Modelling student errors in physics problem-solving

Thesis

UNRESTRICTED

Modelling Student Errors in Physics Problem-Solving

Anthony G. Priest, M.A. (Cantab),

P.G.C.E. (Birmingham)

Submitted in accordance with the requirements of the degree of Doctor of Philosophy at the Open University in the disciplines of Artificial Intelligence and Educational Technology.

October 27th 1986

Date of submission: 27 October 1986

Date of award: 2 July 1987

Acknowledgements

I wish to acknowledge the help and support of Dr. T. O'Shea and Dr. R. Young in the preparation of this thesis, and also the practical and generous assistance of Dr. A. Bundy. Data collection was greatly facilitated by the kind help of Messrs C. Crofts, J. Pritchard and N. Gartside and their students at South Kent College of Technology. Thanks are also due to my wife Judy for her prolonged tolerance and supportive attitude over an extended period.

Abstract

The motivation for this work has been the development of knowledge about the behaviour of human problem-solvers that would enable an intelligent machine tutor to be designed. In the domain of Newtonian Mechanics, this breaks down into two necessary sub-tasks: how do people decide what equation to generate; and what do they produce when they do try to generate an equation? Although these are psychologically separate questions, an automatic tutor for the domain would need to make use of both kinds of knowledge.

Therefore, strategies for controlling search in physics problem-solving are investigated, and a computational model of erroneous solutions is described. Experimental data is used to evaluate the model. Errors in the domain are classified, and the behaviour of problem-solvers predicted under certain circumstances.

Prediction of Novice errors is a crucial ability for an intelligent tutorial system, and the error analysis implemented in the NEWT program is the main contribution of this thesis.

The investigation has two principal aims:

(1) To develop a model that allows a student's future behaviour to be predicted from an analysis of his past actions. It is argued that this is a necessary pre-requisite for the construction of an intelligent tutorial system.

(2) To identify the psychological mechanisms used by problem-solvers working in the domain.

The thesis attempts to achieve these aims in two main ways:

(1) A computer program called NEWT has been constructed, which solves problems of Newtonian Mechanics correctly, or in one of a number of erroneous ways. This allows human errors to be matched, classified, and in some cases predicted.

(2) An analysis of published data leads to the formulation of a control strategy termed "planstacking". This is compared to alternative control strategies, and shown to explain existing data more adequately.

The program is evaluated both as a psychological theory, and as a proposed student model for use in a computer-based tutorial system. The NEWT program was developed from the MECHO program written by Bundy, Byrd, Luger, Mellish and Palmer (1979), at the Department of Artificial Intelligence, Edinburgh University. This program was adapted to produce erroneous problem solutions by the inclusion of procedures to implement malrules observed in the domain.

CHAPTERS

I	Introduction	20
	Selection of Tasks	20
	Object-Level and Meta-Level Search (Definition)	22
	Chronology of the Project	23
	Approaches to the Main Aims	25
	Topics to be Considered	27
	The NEWT System as a Model of Errors	32
	Contributions of the Thesis	34
II	Solving Statics Problems	35
	The Domain	35
	Physical Objects and their Properties	38
	Physical Problems Investigated	42

III	Literature Review	56
	Computer-Assisted Learning	58
	Types of Student Model	81
	Malrules (Definition)	87
	Error Modelling in Problem-Solving	94
	Repair Theory (Definition)	96
	Problem-Solving in Physics	102
	Conclusion	103
IV	Control Strategies and Problem Representations for Solving Statics Problems	104
	The Problem Domain	105
	The MECHO Program	116
	Object Context (Definition)	119
	Chi, Feltovich and Glaser	126
	Larkin, McDermott, Simon and Simon	144
	Sketch (Definition)	174
	The ABLE System	185
	The Planstack Model	204
	The Hidden Curriculum Assumption	229
	Problem-Solving and Errors	230

V Methodology of Computational Modelling 234

Structure of the Theory 234

Derivation of the Malrules 239

Training Set and Validation Set (Definition) 239

The Theories and the NEWT Program 240

Criteria for an Evaluative Measure 245

Numeric Value Criterion (Definition) 245

Accuracy Criterion (Definition) 245

Inaccuracy Criterion (Definition) 246

Parsimony Criterion (Definition) 246

Prediction Criterion (Definition) 246

Assumption Criterion (Definition) 246

Evaluation of Micro-Theory Systems 246

Error Fit Measure (Definition) 246

Micro-Theory Evaluation Quotient (Definition) 250

Span of Consistency for NEWT 253

Span of Comparison for NEWT 254

Summary of Evaluation Criteria 257

Hit and Miss (Definition) 258

Other Forms of Evaluation 261

Student Models in Intelligent Tutorial Systems 262

Conclusions 265

VI Design of the NEWT system 267

Selection of Principles and Contexts 268

Generation of Equations 271

Malrules 273

Relationship Between MECHO and NEWT 290

Conclusion 291

VII The Experiment 292

The Subjects 292

Observations on the Data 295

Identification of New Malrules 297

Description of the Data 299

Hierarchies of Malrule Consistency 302

Relation of Questions to Malrules 306

Summary of Results 308

Analyses Used 312

Analysis of the Use of Lami's Theorem 331

Summary of Analyses 337

Effects of New Malrules 338

Alternative Predictions for the Student Model 339

Conclusions 345

VIII Conclusions 347

Aims of the Project 347

Claims of the Project 348

Identification of Psychological Mechanisms 350

The NEWT Model of Problem-Solving 353

Summary of Results 355

Malrules Identified 357

Limitations of the Project 359

Experimental Tests of Control Strategies 361

Further Work 374

Appendix I

Commented NEWT Code 379

Appendix II

Experimental Data 394

Appendix III

Sample NEWT Traces 424

References 453

Chapter Abstracts

I Introduction

This chapter gives an overview of the project. It motivates the selection of the tasks undertaken and describes the contributions of the work to the fields of Artificial Intelligence and Educational Technology.

II Solving Statics Problems

This chapter delineates the problem domain, and possible methods for solving problems in it. The problems used in the experimental study are stated in English, and in the internal representation used by NEWT.

III Literature Review

This chapter reviews the relevant literature on Computer Assisted Learning and Error Generation in procedural skills.

Representation of erroneous procedural knowledge is identified as a crucial component of an effective student model in a tutorial program, and attempts to develop principled representations of erroneous student knowledge are examined.

IV Control Strategies and Problem Representation for Solving Statics Problems

Control strategies and knowledge representation formalisms used by other researchers are examined, and compared to published experimental results.

A control strategy termed "planstacking" is introduced and defined, and compared to other control strategies. Other outstanding and hitherto unexplained observations are related to a proposed mechanism called "sketch construction", and to a principle termed the "hidden curriculum principle of non-redundancy". These are required to explain some of the observations of initial problem pre-processing by problem-solvers.

V Methodology of Computational Modelling

This chapter looks at possible relationships between psychological descriptions and computational models that implement them.

Three distinct psychological frameworks are proposed for the analysis of problem-solving in the domain of statics:

- a procedural framework for competent problem-solving.

- a procedural framework for consistently erroneous problem-solving.

- a procedural framework for inconsistently erroneous problem-solving.

The relationship of each of these to the NEWT program is described. The abstract relationship between computational models and experimental data is investigated. The inadequacy of conventional statistical techniques is demonstrated, and necessary criteria for an adequate numerical measure of experimental adequacy are defined.

A numerical measure applicable to the domain is proposed, called the "micro-theory evaluation quotient". It is compared to the proposed criteria.

The proposed methods of evaluating the three psychological frameworks and the use of NEWT as a student model are explained and justified.

VI Design of the NEWT System

The top-level control algorithm of NEWT is defined, and its relation to the output (in the form of a list of equations), explained. The three procedural frameworks implemented by NEWT are defined; and a list of "malrules", or erroneous problem-solving procedures, is given. The malrules are categorised into five categories on the basis of their relationship to the structure of the problem-solving process.

The procedural framework for competent problem-solving is related to NEWT without any malrules.

The procedural framework for consistently erroneous problem-solving is related to NEWT peturbed by the addition of one or more malrules.

The procedural framework for inconsistent problem-solving is related to NEWT peturbed by the absence of an appropriate procedure at a particular level of the program.

VII The Experiment

The results of an experimental study are presented in detail. This data is compared to the output of the NEWT system, modified appropriately.

Evaluation measures are presented for each of the three psychological frameworks and for NEWT as a student model.

VIII Conclusions

New ideas have been proposed to explain the control strategy and problem representation of problem-solvers. These have been related to existing data, but further experimental work is necessary to verify them.

Several points of the space of possible errors in Statics problem-solving have been mapped, but more remain to be discovered. The genesis of these errors remains a mystery.

A program has been written which has demonstrated its ability to act as the basis for a student model, and which implements three psychological frameworks for procedural problem-solving. It is better at predicting some kinds of behaviour than explaining it in detail.

The limitations of NEWT are mentioned and avenues for further research described.

Illustrations

Figure 1	The Strut Problem	43, 284, 425
Figure 2	The Unsliding Block Problem	46, 436
Figure 3	The Slope Problem	49, 285
Figure 4	The Angle of Friction Problem	52, 286
Figure 5	The Pulley Problem	195

Chapter I - Introduction

This Chapter gives an overview of the project. It motivates the selection of tasks undertaken, and describes the contributions of the work to the fields of Artificial Intelligence and Educational Technology.

Selection of Tasks

A survey of the literature on Computer-Assisted learning, presented later on in Chapter Three, leads to the conclusion that effective machine tutors necessarily require some form of detailed analysis of the person using them, or 'student model'. This, it is argued, must include all the knowledge necessary to select and implement whatever tutorial strategies are available to the system.

The implementation of such a system implies the selection of a domain satisfying the following requirements:

the domain should be formally taught in the educational system

the domain should be non-trivial

the problems of knowledge representation should not be intractable

These properties are satisfied by the subject of Newtonian Mechanics. It is taught in schools and colleges, it is non-trivial (many students find it insuperably difficult), and it possesses a well-developed formal representation which can be handled with reasonable ease. For these reasons, it is an appropriate domain to choose for investigation.

Tutors in this domain need to be able to identify and correct two kinds of non-optimal actions on the part of students:

-generation of incorrect equations

-selection of irrelevant plans for generating equations.

In the terminology of Bundy et al. (1979), the generation of equations is an "object-level" operation, and the selection of plans is a "meta-level" operation. That is, the structures manipulated by search at the meta-level are themselves operators in the object-level search space. The object-level structures are the elements in terms of which the final solution plan is given.

Of course, the ability to correct non-optimal actions on the part of the student implies an ability to recognise correct equations and relevant plans when they occur. Without the capacity to do these things, the tutor is severely restricted as to the range of tutorial strategies that can be implemented.

Debate still continues as to the most effective tutorial strategy, and therefore progress in the development of intelligent tutorial systems depends on the existence of programs which can identify errors at the object level and also the control strategies used by the problem-solver.

Chronology of the Project

The development of this thesis took place in approximately the following stages:

1979 Literature survey in the field of Computer-Assisted learning, leading to the conclusion that sophisticated student models were a necessity for further progress. Selection of Newtonian Mechanics as an appropriate domain, largely influenced by the MECHO project at Edinburgh.

1980 Visited Edinburgh and used MECHO. Collected some experimental data on Physics problem-solving, and started to consider how errors could be represented procedurally.

1981 Collected more experimental data, and read about the work of Larkin, McDermott, Simon and Simon (1980), dealing with control strategies in Physics problem-solving, and the work of Sleeman (1981) on representing errors as malrules in the algebra domain.

1982 Analysed the errors in the collected scripts in terms of malrules, and started to implement them as additions to the MECHO program. Read the paper by Chi, Feltovich and Glaser (1981) about control strategies.

1983 Visited Edinburgh University and completed the additions to MECHO corresponding to the observed malrules. Started to consider the issues of methodology in validating the model against the experimental data.

1984 Analysed the reported conclusions of Larkin et al and Chi et al; resulting in the formulation of alternative hypotheses for control strategy, for explaining the interleaved nature of problem interpretation and problem solution, and for the mechanisms used in interpreting questions about impossible object configurations.

1985 Defined the micro-theory evaluation quotient and completed the data analysis. Started writing up the thesis.

1986 Completed writing up the thesis.

Approaches to the Main Aims of the Thesis

There have been three complementary approaches to the achievement of the aims of this thesis. These were:

Literature survey.

This was necessary to identify the important issues in computer assisted learning, justifying the selection of control strategy and equation errors as relevant and appropriate areas to focus on. It also yielded a number of experimental results and hypotheses by other experimenters, which are analysed at length later on. It would not have been practically feasible to have considered all the areas looked at in this thesis experimentally, and hence some of the results are derived from the literature analysis.

Collection of Experimental Data

The decision to deal with the analysis of errors in Physics problem-solving led to a need to identify typical errors in the domain. This had to be done experimentally, since no published analysis existed.

Computational Implementation

The guarantee of the completeness of a theory of procedural skill is its computational implementation. Otherwise, as Winston (1984) says, "Much difficulty may be lying under a rug somewhere". The NEWT program shows that the malrules identified in the data analysis are in fact capable of generating the behaviour ascribed to them. Such an approach introduces problems of validation which will be considered in Chapter V.

Topics to be Considered

There are four topics which will be considered in this thesis. A consideration of the literature on Student Modelling will lead to the related areas of intermediate knowledge representation, control strategy, and erroneous behaviour. Existing work in these areas will be related to the aims of the thesis, and to the results put forward in later chapters.

Student Models in Computer-Assisted Learning

Several investigations into Computer-Assisted Learning described in Chapter III have investigated the importance of a psychologically accurate representation of the skill to be learned, in order to give effective tutorial interaction.

This thesis focusses on the learning of one specific skill- the solution of problems in Newtonian Mechanics- and concentrates on elucidating the psychological mechanisms used by novice and expert problem-solvers, with the intention of developing results relevant to the construction of a computer-based tutor for this domain.

Considerable work has been done on the psychological methods and representations used to solve problems in this domain (Hinsley, Hayes and Simon 1977; Larkin, McDermott, Simon and Simon 1979, 1980; Chi, Feltovich and Glaser 1981), and in Chapter IV this material will be examined in depth.

In these papers, data has been reported which places strong constraints on any theory of psychological representation one may wish to put forward. However, there is little work as yet that reflects directly on one issue of major importance for any computer-based tutor: the nature and generation of errors.

Existing theories deal at some length with intermediate knowledge representations and with control strategy, but contain little or nothing that would explain erroneous behaviour by problem-solvers.

Intermediate Knowledge Representations

Larkin, McDermott, Simon and Simon (1979, 1980), have postulated the existence of a number of intermediate knowledge representations used by the problem-solver. These will be considered in Chapter V, together with the experimental data they relate to. Other workers, such as Chi, Feltovich and Glaser (1981) infer the existence of such internalised knowledge structures, but have not elaborated their theory to the extent that unambiguous predictions can be derived from it.

An intermediate knowledge structure called a "sketch" is put forward in Chapter IV as a possible explanation for some of the results found by other workers, especially Paige and Simon (1966). This structure closely resembles the class of "spatial mental models " described by Johnson-Laird (1983).

Control Strategy

A variety of mutually exclusive control strategies have been put forward to explain the experimental data at the level of order of equation generation. These include:

- Backward Inference (Marples 1974)
  based on an analysis of sought quantities

- Forward Inference (Larkin 1980)
  based on object recognition

- Forward Inference (Chi, Feltovich and Glaser 1981)
  based on schemata corresponding to physical principles

in this thesis, the existing data is re-analysed to yield support for a fourth control strategy:

- generation of equations from plans generated by backward inference and stored on a push-down stack.

This strategy is here termed "Planstacking", and is compared for explanatory adequacy to the alternative strategies in Chapter IV.

Erroneous Behaviour

Novice problem-solvers frequently produce erroneous behaviour, although the errors they evince have not been explicitly taught.

Larkin (1980), describes a computational model which can duplicate a small proportion of the observed errors, but which is inherently incapable of generating others. No other work in this domain has put forward any explanation for erroneous behaviour.

Errors have, however, been extensively studied in the domain of arithmetic (Van Lehn 1981; Young and O'Shea 1981; Brown and Burton 1978) and linear algebra (Matz 1982; Sleeman and Smith 1981).

One approach to the explanation of errors is to propose the existence of consistent but incorrect procedures termed "malrules", which generate erroneous behaviour. The malrules of a domain bear a generic resemblance to correct rules, and have been extensively studied by Sleeman (1984), and by Sleeman and Smith (1981) for the domain of equation solution. This is the approach taken in this thesis: it is assumed initially that errors are caused by incorrect but well-defined

computational procedures whose nature can be inferred from the student's output. Since no other researchers have published data on errors in this domain, a study was undertaken to identify malrules in the work of novice subjects.

The NEWT System as a Model of Erroneous Problem-Solving

Malrules collected from scripts of novice problem-solvers were included in a problem-solving program called NEWT, which was adapted from the MECHO program (Bundy, Byrd, Luger, Mellish and Palmer 1979); and this was compared to the behaviour of a second set of students solving problems of Newtonian Mechanics.

It was found that six malrules were sufficient to account for about 70% of the consistent errors made by students, but that there was evidence that not all erroneous behaviour could be explained in this way. In addition to stable erroneous procedures, many students appeared to posses some mechanism which caused erroneous solutions to be produced which varied in an inconsistent and unpredictable way. Since the essence of a malrule is its coherence and stability, such behaviour is inherently impossible to explain in terms of malrules.

The explanation of errors in terms of malrules unifies many different observed behaviours by the introduction of a small number of well-defined processes, but this is still only an intermediate level of explanation. The question of how malrules arise (and indeed, how they demise also) is not addressed in this thesis, but remains an area for future investigation. Priest (1987) has put forward some speculative hypotheses to account for malrule generation, based on the 'Repair Theory' concept of Van Lehn (1981); but these have not yet been tested experimentally, nor given a computational implementation.

It is also relevant to note here that the theories examined in this thesis are theories of behaviour, and do not have anything to say about the mental processes of students who do not produce any behaviour. Such students remain outside the scope of all the theories here examined.

Contributions of the Thesis

This thesis makes contributions to the fields of Artificial Intelligence and Educational Technology. Its main contribution to Artificial Intelligence is the definition and study of a new control algorithm suitable for solving problems in Newtonian Mechanics, or other similar domains. It also contributes to the field by considering the issues of validating a computational model against observed behaviour, for which purpose a measure termed the "micro-theory evaluation quotient" has been defined and used.

The main contribution of this work to Educational Technology is an analysis of errors in the domain of Newtonian Mechanics. The errors have not only been identified, but classified as to type, consistency and frequency; and the accuracy of their formulation has been verified by a computational implementation which demonstrates the extent to which a machine tutor may be expected to be able to predict novice behaviour. This data lays the basis on which an effective tutor in the domain may be constructed.

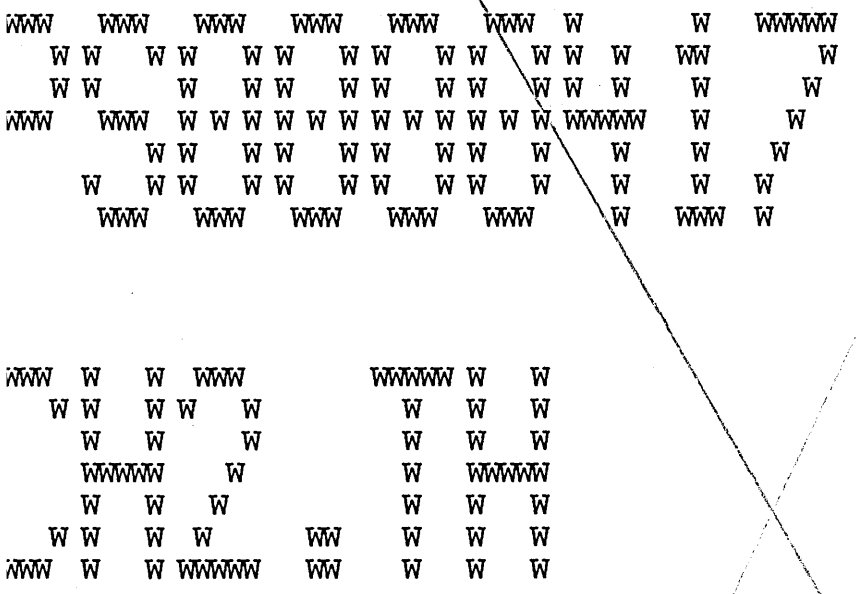
BOUND IN ERROR

£1·10

ser: PS0000417 -at LASER

h2.th

****
****



****
****

abel: PRT003 -form

pened from: <SYSF01>MSCR>PS0000417>THESIS.DIR

pooled: 87-09-10.09:47:48.Thu [Spooler rev 19.4c]
tarted: 87-09-10.11:29:12.Thu on: AMLC by: LASER

Laser Printer 2

Chapter II - Solving Statics Problems

This chapter delineates the problem domain, and possible methods for solving problems in it. The problems used in the experimental study are stated in English and in the internal representation used by NEWT.

The Domain

The problem-solving domain investigated in this thesis is that of statical equilibrium in Newtonian Mechanics. This concerns the physical relationships involved in situations involving idealized objects such as rigid beams, light inextensible strings, points with and without mass, surfaces with and without friction, springs that obey Hooke's law, and so on.

From now on, the characters "mu" and "lambda" will be used to stand for the Greek letters of the same name. These symbols are used to denote the coefficient of friction and the coefficient of elasticity respectively. The convention used in this thesis will be that concatenated groups of symbols will be

used to indicate single quantities. The multiplication operator will be represented explicitly as an asterisk.

Physical Principles Implemented in the NEWT System

A-level questions in this domain involve finding one or more physical quantities such as a particular force, length, angle, or mass from a description of a configuration of physical objects and a knowledge of some of the objects involved in the problem.

The physical principles applicable to the domain are:

- Resolution of forces (Newton's Second Law)
- Principle of Moments
- The Friction Law (F = mu * R)
- Hooke's Law (tension = lambda * extension)

It was decided not to investigate problems involving Hooke's Law, since this was often taught separately from the rest of the principles mentioned to the students available to the study. It was therefore difficult in practice to obtain a

large enough sample who had covered Hooke's Law as well as the other physical principles.

Some work was done on the investigation of errors involving the Principle of Moments. It eventually became apparent that a significant number of errors involving the Principle of Moments depended on the subject assuming a sense for the magnitude of a force vector. This corresponded to the direction of the arrowhead marked on the diagram of the problem. In cases where the assumed sense of the force was incorrect, certain errors were made that depended on the fact that the assumed and actual senses were different.

To model such an error computationally, it is necessary for the program to maintain a representation of forces that corresponds closely enough to that of the subject to have an "assumed sense". The method used by MECHO to represent forces defines the line of a force, but does not specify the assumed sense of the force with respect to this line of action. Altering MECHO to do this would require extensive rewriting of all procedures that handle vector quantities- essentially rebuilding the system from scratch.

Therefore the malrule investigations of NEWT have been confined to the subdomain of statical equilibrium defined by the principles:

Resolution of Forces

Friction

Physical Objects and their Properties

The physical objects which NEWT can represent in a problem configuration are those corresponding to the following idealized objects:

Fixed point - does not move, and has no size. if any force acts on it, a force of reaction keeps it stationary.

Point - has no size or mass. Points can be used to label parts of bodies

Particle - has mass but no size. Gravity acts upon it, and it can be attached to other objects so that forces can be transmitted to it.

Rod - has a (possibly zero) mass and a length. The rod can transmit forces of tension or compression, and also transmit moments from one end to the other. The idealized rods used in these problems never break or bend, even though they have no thickness. A rod has two endpoints, which are points, and it has an inclination defined by the vector separating its endpoints. Other points may also be defined along a rod.

String - A string is light, inextensible, and has no thickness or stiffness. It is not hairy. It can exert a force of tension, but cannot transmit moments. A string has two endpoints, and other points along it may also be defined.

Surface - A surface is fixed, and may be smooth or rough. If it is smooth, no force of friction acts on it, only forces of reaction are possible. If it is rough, then the friction law F <= mu * R can be applied to bodies resting on it.

These principles and ideal objects define the domain in which NEWT operates. There is no limit to the complexity of the problems that can be solved within these limitations.

As well as the principles defined above, NEWT will take the following things 'for granted', when generating equations:

- If a body has a mass M, then a force M * g acts on it in a downward direction. This does not apply to fixed bodies, which are regarded as part of the ground.

- If two things are in contact, a force of reaction acts between them. If one of the things is a surface, the force of reaction is normal to the surface. This does not apply to strings, for which the concept of 'reaction' is included in 'tension'.

- any string is automatically assumed to have a tension. This may be zero, but cannot be negative.

- objects attached to each other may transmit forces to each other.

MECHO possesses a well developed mechanism for listing the forces acting on an object, and summing their components in a given direction. There is no inbuilt limitation to the number of forces which may act at a given point.

Physical Problems Investigated

In the initial stages of the project a number of problems were investigated; while in the final studies of student problem-solvers, four particular questions were considered. These will be referred to as:

The Strut

The Unsliding Block

The Slope

The Angle of Friction

Of these four, solutions to the 'unsliding block', problem have been investigated by Larkin (Larkin, 1980). The other problems do not appear to have been investigated psychologically before.

These problems are now presented in both their natural language forms, and in the predicate calculus formalism used by NEWT.

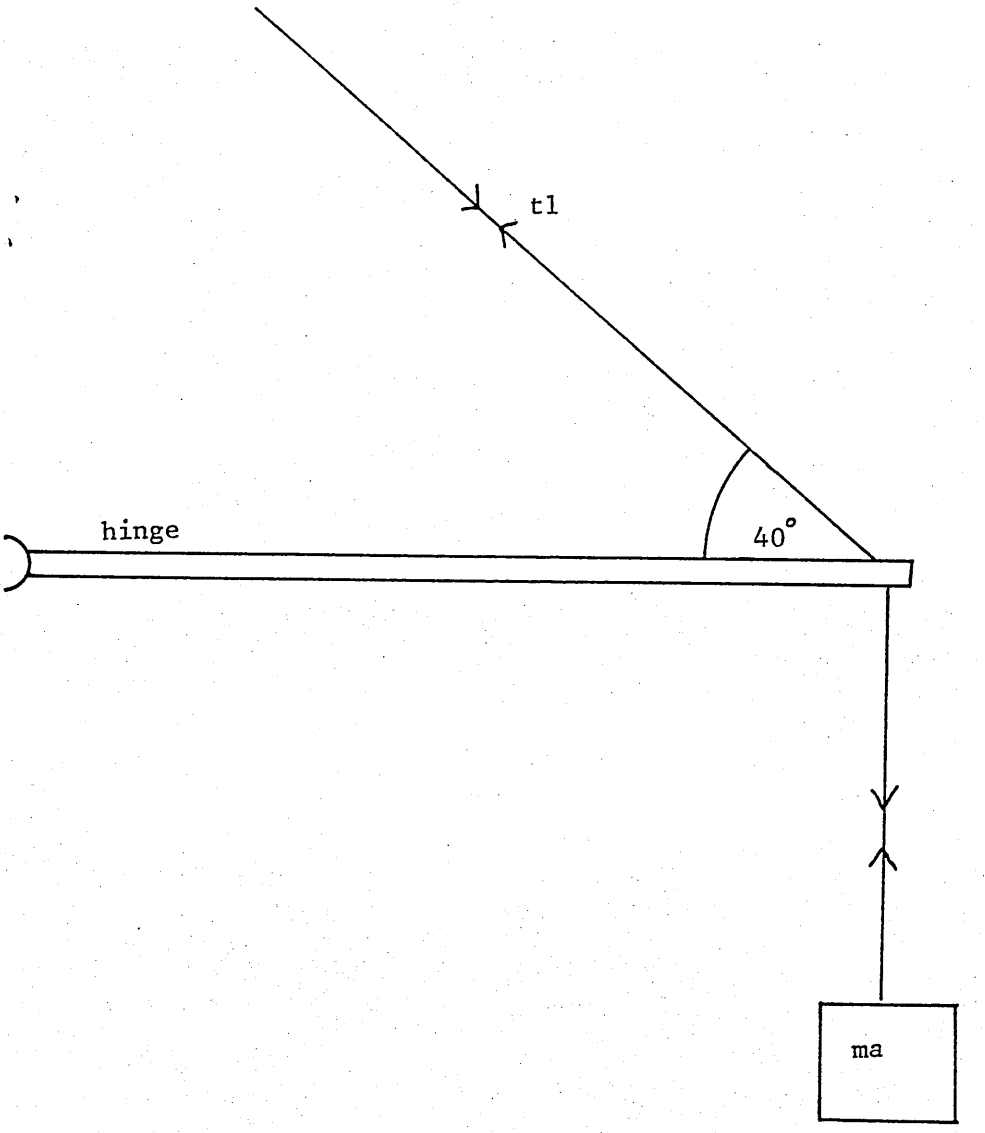


Figure 1. The Strut Problem

A particle of mass Ma hangs from the end of a horizontal strut whose other end is hinged to a wall. The strut is supported by a string making an angle of 40 degrees to the strut. Find the tension in the inclined string.

The internal representation of this problem used by NEWT is as follows:-

```
isa ( period, now ).
isa ( particle, a ).
cue linesys ( string,strut, [lend, rend] ).
cue linesys ( string, string1, [top1, bottom1] ).
cue linesys ( string, string2, [top2, bottom2] ).
concavity ( strut, stline ).
concavity ( string1, stline ).
concavity ( string2, stline ).
tangent ( strut, 180 ).
tangent ( string1, 140 ).
tangent ( string2, 90 ).
fixed-contact ( rend, bottom1, now ).
fixed-contact ( rend, top2, now ).
fixed-contact ( a, bottom2, now ).
fixed ( lend, now ).
```

```
fixed ( top1, now ).
tension ( string1, t1, now ).
mass ( a, ma, now ).
given ( ma ).
sought ( t1 ).
:- assert ( minimal-part( string, str, Pt, str ) ),
   assert ( static (_,_) ).
```

In this formalism, the constant "now", is introduced because MECHO is capable of dealing with dynamic behaviour. In statics problems, this becomes redundant. The "cue" lines indicate that when the problem is read in, a schema is automatically used. When MECHO is told that an object is a string or a rod, it immediately makes the "common sense", inference that the object has two ends, which it creates names for if they are not already specified in the question.

The final line of the problem states that all objects are static, and the line before it asserts that none of the strings needs to be considered as made up of two other strings joined together. When dealing with questions about strings passing over pulleys, MECHO needs to be able to conceptualise the string as consisting of two separate strings, one each side of the pulley. In this case, such a reconceptualisation is not required.

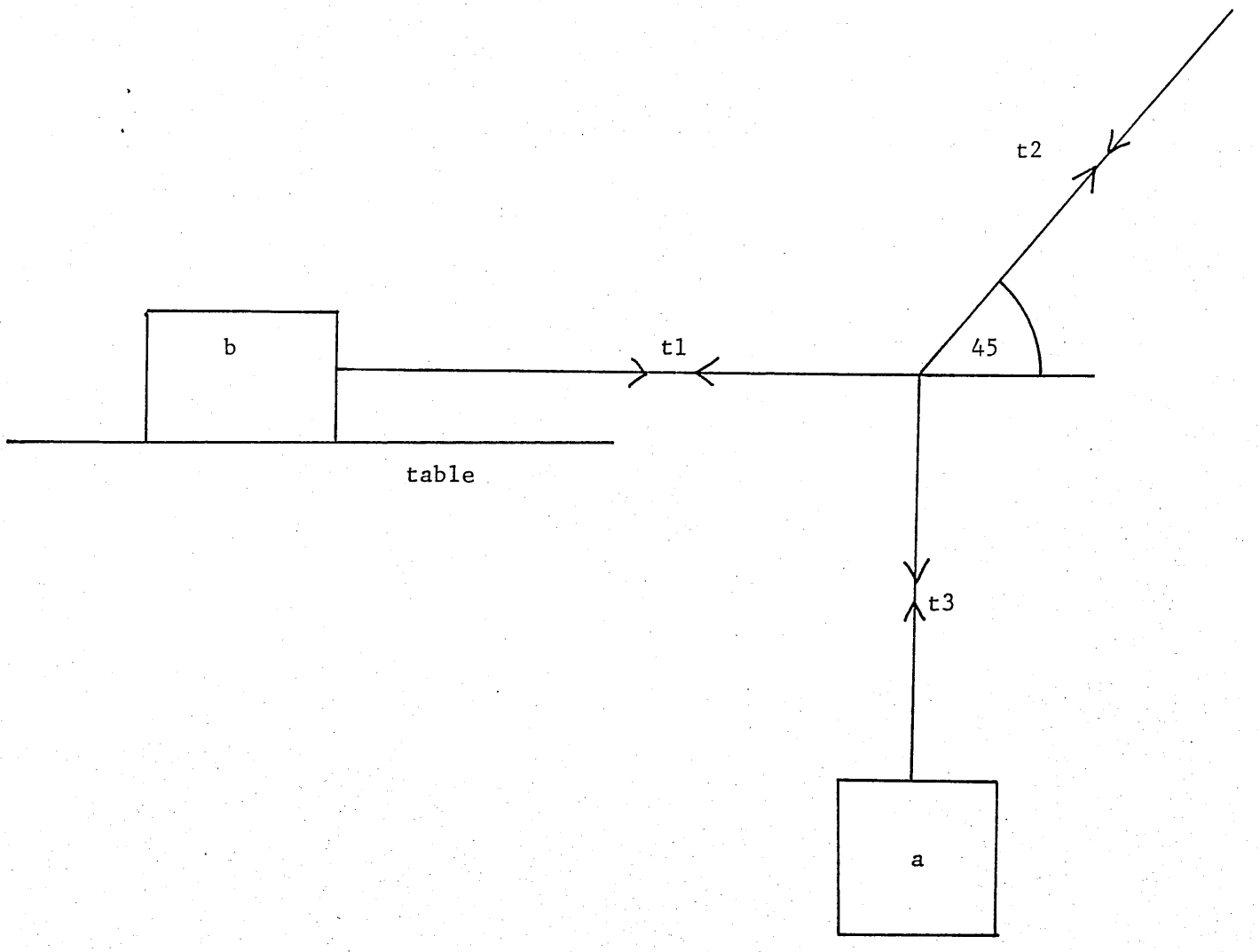


Figure 2. The Unsliding Block Problem

A block of mass mb rests on a rough horizontal plane, whose coefficient of friction is mu, the block being in limiting equilibrium. The block is pulled by a horizontal string attached to two other strings, one at an angle of 45 degrees to the horizontal whose other end is fixed, and one hanging vertically. On the other end of the vertical string a mass of ma is suspended. Find ma.

The internal NEWT representation is:

```
isa ( period, now ).
isa ( particle, a ).
isa ( particle, b ).
cue linesys ( path, table, [lend, pt/moving, rend] ).
cue linesys ( string, string1, [pt1, pt2] ).
cue linesys ( string, string2, [pt3, pt4] ).
cue linesys ( string, string3, [pt5, pt6] ).
concavity ( table, stline ).
concavity ( string1, stline ).
concavity ( string2, stline ).
concavity ( string3, stline ).
tangent ( table, 180 ).
tangent ( string1, 180 ).
tangent ( string2, 225 ).
tangent ( string3, 90 ).
```

```
fixed-contact ( b, pt, now ).
fixed-contact ( b, pt1, now ).
fixed-contact ( pt2, pt3, now ).
fixed-contact ( pt2, pt5, now ).
fixed-contact ( pt6, a, now ).
fixed ( pt4, now ).
static ( a, now ).
static ( b, now ).
wh-side ( b, table, left, now ).
mass ( a, ma, now ).
mass ( b, mb, now ).
coeff ( table, mu ).
solid ( table ).
given ( mb ).
given ( mu ).
sought ( ma ).
```

In this representation of the problem, the expression "pt/moving", refers to the body being in limiting equilibrium, and carries no implications of actual movement. The statement "wh-side (b, table, left, now)." is used to conclude that the body is above the table rather than below it, and so will not fall off. The "left", is interpreted with regard to the tangential direction of the table to infer this.

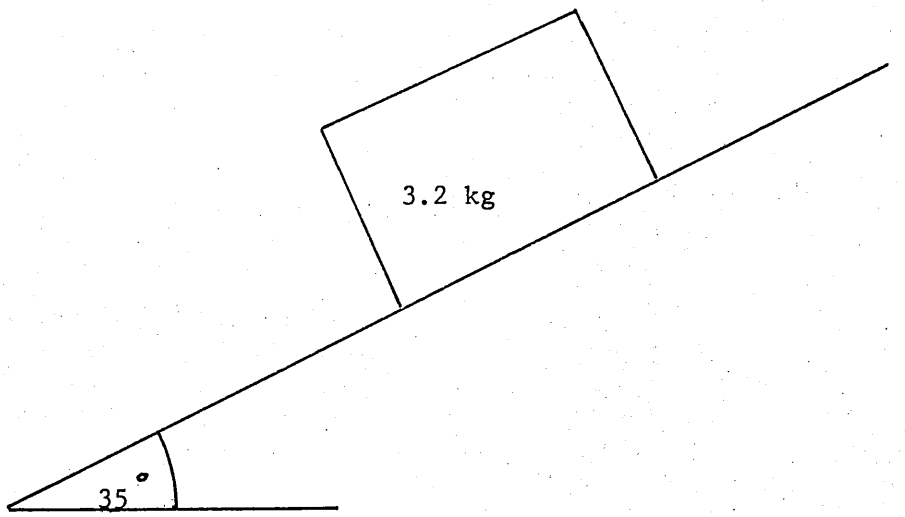


Figure 3 . The Slope Problem

A block of mass 3.2 kg in equilibrium on a rough plane sloping at 35 degrees to the horizontal. Find the force of friction, f, and the force of normal reaction between the plane and the block.

The internal representation of this is:

```
isa ( period, now ).
isa ( particle, body ).
mass ( body, mb, now ).
measure ( mb, '3.2', kg ).
solid ( slope ).
concavity ( slope, stline ).
tangent ( slope, 215 ).
incline ( slope, 215, pt ).
:- assert ( current-period ( now ) ).
wh-side ( body, slope, left, now ).
static ( body, now ).
static ( slope, now ).
cue linesys ( path, slope, [lend, pt, rend] ).
fixed-contact ( body, pt, now ).
coeff ( slope, roughness ).
```

No quantities are marked as "sought", because no names have been assigned to the desired quantities in this problem. If NEWT is instructed to resolve forces parallel to and perpendicular to the slope, it will generate names for the desired quantities, and also use physical principles to generate equations to find their values.

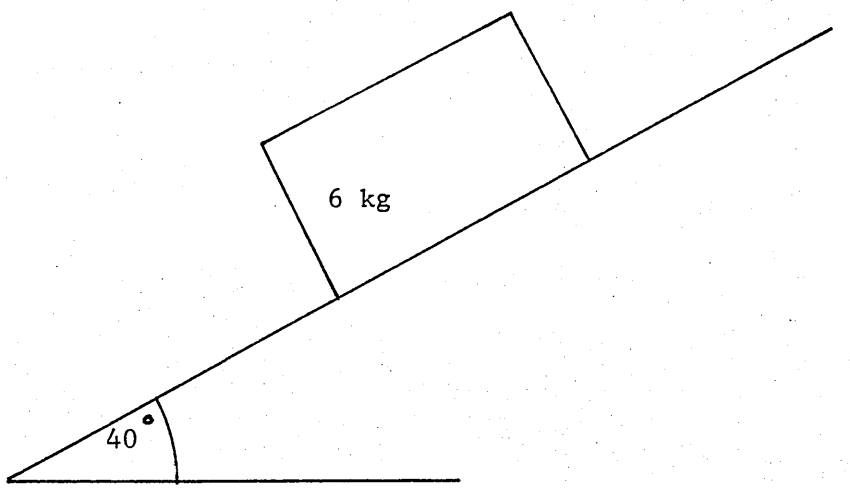


Figure 4. The Angle of Friction Problem

A block of mass 6 kg. lies on a rough plane inclined at an angle of 40 degrees to the horizontal. If the block is in limiting equilibrium, find the coefficient of friction between the block and the plane.

The internal representation is:

```
isa( period,now ).
isa( particle,block ).
cue linesys( path,plane,[lend,pt/moving,rend] ).
concavity( plane,stline ).
incline( plane,40,pt ).
static( block,now ).
wh_side( block,plane,left,now ).
mass( block,mb,now ).
measure( mb,6,kg ).
coeff( plane,mu ).
solid( plane ).
fixed_contact( block,pt,now ).
given( mb ).
sought( mu ).
:-assert( static(_,_) ).
```

Example Solution

Here is an example solution of the strut problem. It requires the application of the principle "Resolution of Forces", to produce a single equation containing the sought quantity and various known quantities.

The problem-solving process described in this thesis halts at the point where an equation with only one unknown quantity is produced, on the assumption that this equation can be solved.

Sought quantity: t1

Resolving vertically at "rend":

t1 * sin 40 = ma * g

In this example, "t1 * sin 40" is the upward component of the tension in the supporting string. "ma * g" is the weight of mass "a". Since no other force acts in a vertical direction on the right-hand end of the strut, these forces must be equal. This yields the equation from which t1 may be found.

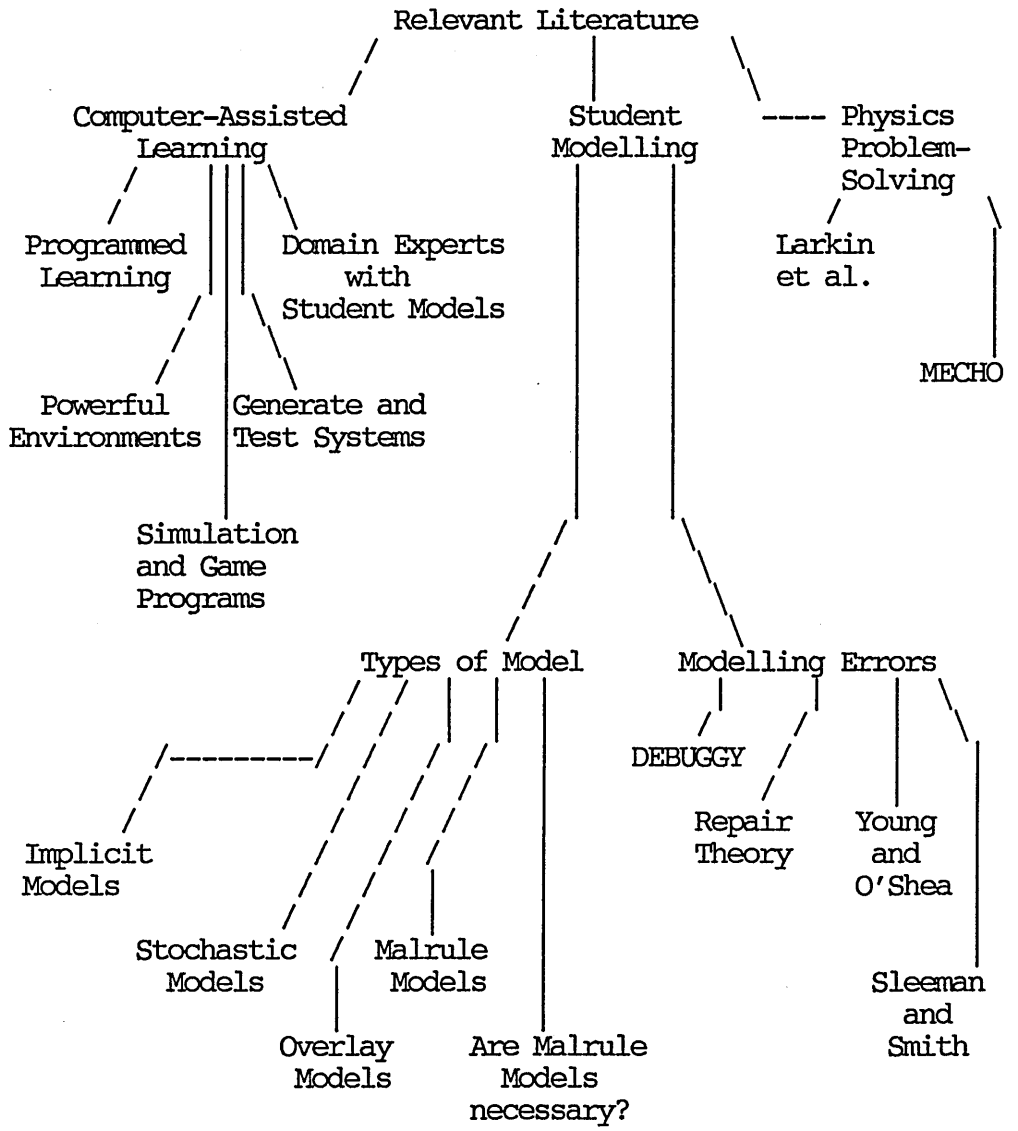
Relative Difficulty of Problems

It may seem at first sight that this problem is too simple to yield any interesting behaviour or consistently erroneous procedures. Not a bit of it- the students in the experimental sample often found this problem impossible to solve unaided. The hardest of the test problems is undoubtedly the "unsliding block" problem, which was successfully solved by the smallest proportion of subjects of any of the problems.

Chapter III - Literature Review

This chapter examines the literature relevant to the development of computer-assisted learning and the psychological modelling of human problem-solving.

The chapter is structured as follows:



Computer-Assisted Learning

Computer-assisted Learning (CAL), has a history almost as old as the computer, and is currently undergoing a revival with the widespread distribution of programs aimed at microcomputers.

There are five main approaches to using the computer as a learning aid:

(1) Programmed Learning (Author Languages)

(2) Powerful Environments

(3) Simulation and Game Programs

(4) Generate-and-Test systems

(5) Domain Experts with Student Models.

Programmed Learning

This is the oldest CAL approach, deriving from the programmed learning textbooks and "teaching machines", that were first developed in the 1950s (Annett 1966).

The psychological basis for this approach was originally the behaviourist psychology of Watson and Skinner (Skinner 1954) but later practitioners of programmed learning have adopted a more pragmatic approach. Essentially, a programmed learning program (or text), is a series of unitary screen displays called "frames"; after each of which, the user is required to answer a question in a format which only allows a restricted range of answers, such as multi-choice questions.

Depending on the answer given, a different frame will follow next, until the user has completed the prepared teaching sequence. Such programs are conceptually simple to design, although they require considerable care and detailed planning in organising and selecting the "frames" to be displayed. Examples of large- scale developments along these lines are the PLATO (PLATO IV Software Group 1974) and TICCIT (Bunderson

1974) systems, both of which were designed to run on special-purpose hardware, both of which were originally designed to replace rather than supplement human teachers.

The main difference between the two systems from the educational point of view is that whilst TICCIT lessons were prepared by teams involving professionals from a variety of backgrounds, the PLATO project relied from the first on practising teachers producing their own teaching material. Consequently, there is a vast amount of PLATO teaching software now written, some of better quality than others.

The results of these two large-scale projects have been disappointing when viewed in the light of their original projections:

- Programmed Learning courses have not been able to replace live teachers to any significant extent. They therefore form an add-on cost to conventional instruction rather than a cut-priced alternative, or even a high-priced alternative (Hooper 1975).

- Classes taught exclusively by such teaching programs rarely achieve better results than control classes, and frequently suffer from higher drop-out rates (O'Shea and Self 1983).

Since the inception of the PLATO and TICCIT projects, many, many programmed learning programs have been written for use on standard hardware, and similar conclusions apply to them as well (Self 1974).

In order to facilitate the swift and easy production of frame-based teaching programs, a number of special-purpose languages, such as TUTOR (Ghesquiere, Davies and Thompson 1974) and COURSEWRITER (IBM Corporation 1971), have been developed. These have enabled computer-naive subject experts to produce programmed learning programs faster, but not to produce fundamentally different systems (Sleeman and Brown 1982, see introduction).

The limitations on greater effectiveness for this type of teaching seem to be twofold:

(1) The program does not "understand" its subject matter in any way.

(2) The program does not usually have any idea of what the learner does or doesn't know. Attempts to index topics learnt via flags set by answers to particular frames have proved cumbersome and inflexible to use.

Any apparent "understanding" exhibited by the program is the result of careful forethought by the designer of the frames displayed. He is required to predict in advance what answers a user will make to the questions, deduce from this what the user's state of knowledge is, (or will be), and what action the program should take from that point on. Not only is this a very difficult thing to do, especially when one only has such a coarse instrument as a multi-choice question to diagnose errors by, but the task is in principle an enormous one relative to the size of the teaching sequence.

This follows from the fact that if at each stage of a sequence there are N possible choices, then the number of possible frames needed to allow for every conceivable path through a teaching sequence of X frames is:

Stage	No of Frames at this Stage
1st	1
2nd	N
3rd	N^2
.	
.	
.	
Xth	$N^{(X-1)}$

$$1 + N + N^2 + \ldots + N^{(X-1)} = N^X - 1$$

For example, let us suppose we have a teaching program in which every frame can lead to four possible student responses. Then the number of frames required is:

Length of teaching sequence in Stages	No. of frames
5	1023
10	1,048,575
15	1,073,741,823
20	1,099,511,627,775

Even with a limited branching rate (and hence a rather coarse information feedback mechanism), the number of frames needed becomes unmanageably large after even a short lesson sequence. If an author were able to complete a frame in ten minutes, then it would take eighty-five years to complete the frames required for a ten-lesson sequence (assuming the author works forty hours a week for fifty weeks a year). The disproportionate

increase in workload compared with increase in the task undertaken is termed "combinatorial explosion", and is an insuperable difficulty for any system based on pre-stored responses to student actions.

Powerful Environments

A radically different approach to the use of computers in education is to provide an interactive computing environment in which the user can investigate the space of possible actions available to them. The philosophical underpinning of this is the concept that people learn more by doing than by being done to (Piaget 1971).

Projects taking this approach have varied in the extent to which the student's actions are structured. At one extreme is the belief that the students should decide entirely for themselves what direction to take, and should not be guided even implicitly in any particular direction. At the other extreme, the interactive environment can be seen as an aid to learning a particular syllabus, and the student can be given explicit instructions as regards what type of problems to solve, and how to go about solving them.

It is essential in this approach that the interactive language should be easy to use; and also that it should be a natural medium for the expression of the kind of concepts that it is designed to deal with. The LOGO language has been developed at MIT (Papert, Watt, diSessa and Weir 1979), and its application to teaching investigated at Edinburgh University (Howe, O'Shea and Plane 1980) as a medium for the expression of programming and problem-solving concepts in general, and graphical concepts in particular. Papert (1980) speaks of "powerful ideas" which develop naturally when a student uses an interactive environment in which they are naturally expressible. These "powerful ideas" are high-level problem-solving concepts hard to define in any specific way, but vital for the development of general problem-solving skills. Examples might be the idea of breaking a problem down into inter-connected subproblems, the ideas of recursion and iteration, or the idea of formally specifying algorithms for problem solution.

The approach to LOGO propounded by researchers at MIT is to introduce the language to children in a non-directive way, in the belief that student-motivated action and interaction will inevitably lead to the development of powerful ideas and in improvement in general problem-solving skills (Papert 1984).

At Edinburgh, a more structured approach was employed, in which LOGO concepts were related to the secondary Mathematics curriculum. Worksheets and exercises were produced, and students were given lessons in the use of LOGO. There was a considerable latitude for student experimentation as well, which supplemented the more directed parts of the student's LOGO experience (Howe, Ross, Johnson, Plane and Inglis 1982).

A major problem with LOGO projects has been the difficulty of evaluating them. The more closely the use of the language has been tied to an existing syllabus, the easier it is to evaluate as a tool in learning that topic. Evaluations conducted at Edinburgh have shown some benefit to student's mathematical abilities, especially to students with learning difficulties (Goldenberg 1979); but it is not clear whether these improvements could not have been obtained by the same amount of effort expended on traditional forms of teaching.

Evaluating unstructured LOGO use by students is virtually impossible, because it is impossible to know what is being claimed. Some people introducing LOGO into education in this way even argue that evaluation is irrelevant or in some way harmful. Papert (Papert 1984) describes the ideal educational environment in the following terms:

> "Society will be able to face the task of inventing environments in which children can develop as social, loving, honest human beings without distorting this goal by the crudely technical one of stuffing the multiplication tables into their heads."

The other language which has been proposed for use as a "powerful environment" is PROLOG (Briggs 1982). This is not designed as a graphically expressive language like LOGO; but its strong points as an educational tool are that it is easy to develop relational databases in PROLOG, and that much of its syntax and semantics are derived from logic (i.e. the predicate calculus). Database query systems are easy to develop in PROLOG, and its similarity to predicate calculus allows the user to develop what some PROLOG users consider to be a powerful idea- the fact that a statement in PROLOG has a

dual interpretation. Any PROLOG statement (or "clause"), can be read "declaratively", as a statement that some relationship is true; or "procedurally", as an instruction to perform some set of operations (Kowalski 1984). Unlike LOGO, PROLOG is also used outside the classroom for writing large programs; but like the work with LOGO, the PROLOG projects are hard to evaluate, and for similar reasons. It is not clear what would constitute success for a scheme that involved teaching PROLOG in schools, unless learning PROLOG were seen as an end in itself.

Simulation and Game Programs

A separate approach to computer assisted learning is given by simulations and game programs.

In a simulation program, some real-life system is modelled by a computer program. The user is able to choose various initial parameters, and see how the model develops, the purpose being to illustrate and explain whatever is being simulated. Such programs are widespread, for example there are simulation programs dealing with atomic and molecular orbitals (Buist

1978), population dynamics (Boyle and Anderson 1978), biological simulations (Murphy 1981), and so on. In many cases, these programs represent things that are difficult, slow, expensive, or impossible to model in laboratory experiments. Unlike programmed learning systems, a simulation program does not necessarily require a "correct answer", from the student- the task of the program is to supply the answers itself.

Evaluating the student's knowledge is not usually the function of such systems, and the choice of different material appropriate to a student's needs may not arise- a simulation is designed to illustrate rather than test.

Computer games may resemble simulations superficially, but they have the added factor that the student can win or lose; and that therefore he can make better or worse moves. This makes it possible in principle to infer the student's knowledge state, and therefore to act in such a way as to refine or extend it.

Two examples of computer games intended to be used in a CAL context, (rather than as amusement arcade novelties), are WEST (Burton and Brown 1982); and WUMPUS (Goldstein 1982).

The WEST program sets up an interactive game for the user called "How the West was Won". In this, a pictorial representation of a winding road appears, and the user has to conduct a "stagecoach" from one end to the other, taking advantage of short cuts if possible.

In every move, the user is presented with three randomly chosen integers, and is required to make a "move" corresponding to the result of arithmetical operations on the numbers. For instance, if the numbers available were 3, 2 and 5; he could select the move values:

```
( 3 + 5 ) / 2  =  4

    3 * 2 - 5  =  1

    3 * 5 + 2  = 17
```

and so on.

When a value is selected, the stagecoach moves forward that number of spaces, and then takes a shortcut if it lands on one. The idea is to familiarise the student with basic arithmetic operations.

Burton and Brown realised that in practice, many students were settling for a particular strategy that sometimes predicted suboptimal moves. Giving students a game and letting them get on with it did not necessarily lead to a full comprehension of the domain. They concluded that a "coach" was necessary which would model the student's knowledge state, and make suggestions and comments as necessary. Knowing when to interject was regarded by Brown and Burton as a difficult problem, whose solution depended on an accurate student model.

Extensive testing of WEST with both real and simulated students showed that it developed accurate student models when the students behaved consistently, and that coached students performed better than uncoached ones. Brown and Burton expressed caution about the possibility of transferring their techniques to more complex worlds because of the limited nature of the WEST environment.

Goldstein's WUMPUS program was designed as a test-bed for theories of knowledge representation and tutoring, rather than for teaching a specific skill. In this game, the player visits various caves in a labyrinth, seeking the deadly Wumpus, and slaying it with arrows if possible. Since the caves are dark as drainpipes, the player can't actually see anything, but is

informed of squeaks (for bats), draughts (caused by pits), and smells (caused by the Wumpus itself). The skill in playing the game consists of being able to evaluate evidence from multiple sources regarding danger, and of inferring the location of the Wumpus.

Absence of indicators such as smells and draughts also needs to be evaluated to infer which caves are safe to enter. Goldstein drew up a procedural representation of the knowledge necessary to play the game, and structured it in the form of a directed graph.

This procedural graph was intended to model not only a student's knowledge, but also the growth and development of his strategic knowledge as he learned more sophisticated WUMPUS playing strategies. This is done by representing the student's knowledge at any one time as a subset of the nodes of the Genetic Graph which corresponds to the expert's procedural knowledge.

The goal of the program was firstly to develop an adequate formalism to describe the growth and development of procedural knowledge; and secondly to compare the effectiveness of different tutoring strategies. Goldstein was convinced that a competent tutoring strategy could only be built if it could

make use of a student model. As Goldstein says, "To offer appropriate tutorial advice, a teacher must accurately model the student."

Goldstein did not compare the effects of learning WUMPUS while tutored by his program to the effects of learning WUMPUS from a human tutor. Therefore any evaluation must be tentative. His own conclusions about the effectiveness of the tutorial program based on the genetic graph (referred to as "GG"), were:

> "It provides a more powerful foundation for modelling than either a script of correct answers or a set of expert skills. Nevertheless, the GG does not solve the modelling problem. While the process of constructing a model gains guidance from the graph, it remains complex."

Generate-and-Test Systems

For the teaching of procedural skills such as subtraction or circuit analysis, an alternative to storing large numbers of questions on a file is to generate questions by some process internal to the program, ask the student for his answer, and then compare his answer with that of another process capable of solving the problem. On the basis of this comparison, feedback can be given to the student about his performance. An example of a generate-and-test system which has been developed is SOPHIE, which teaches fault analysis in electronic circuits (Brown, Burton, and deKleer 1982).

Unlike frame-based teaching programs, generate-and-test systems do have an understanding of what they attempt to teach, in the sense that they can solve the problems they expect the student to cope with. Another advantage of such systems is that they are in principle capable of asking an indefinitely large number of questions- they are not limited to a pre-stored list of examples (Palmer and Oldehoeft 1975).

Since the program has to generate problems, solve them, and then compare the student's answer with the computed solution,

such systems require more sophisticated programming than frame-based systems, and their production cannot be streamlined by the use of special purpose languages like the author languages already mentioned. This makes them significantly more expensive, and in many cases too large to run on a small computer.

Although generate-and-test systems are complex from the programming point of view, in some ways they offer less opportunity for incorporating sophisticated educational insights into their behaviour. Since the control and remediation strategies of a frame-based program are inherent in the pre-stored frames, it is possible for an author with a sophisticated understanding of what is being taught to anticipate the most likely errors on the part of the future users of the system, and to incorporate frames intended for the clarification of particular expected errors. This is not possible with a generate-and-test system, because the creator of the system will not know exactly what questions will be asked. The system's reaction to the student's input is therefore more likely to take the form of a simple indication of whether the answer is right or wrong. If the domain is appropriate, the student could be informed of the consequences of his intended actions.

For example, the SOPHIE program simulates the behaviour of an electronic circuit. A fault is chosen, and the behaviour of the faulted circuit calculated. The user can ask for voltage and current readings at various circuit locations, and is expected to work out the nature of the fault.

He can instruct the program to replace components he believes faulty, and it will calculate the consequences of this action. The program can be said to have a good working knowledge of circuit analysis, but the original version had no knowledge of the student, and could not adapt its chosen examples or explanations to the level appropriate to the student's knowledge, (later versions of SOPHIE have been developed in this direction).

Another example of a generate-and-test system is ACE (Sleeman and Hendley 1982), which teaches students to interpret nuclear magnetic resonance spectra. This selects a problem which consists of a formalised numeric representation of a nuclear magnetic resonance spectrum for some chemical. The system has access to a module which is able to determine the chemical structure of the molecule by an algorithm involving qualitative reasoning and backtracking. The student is then asked to make statements about the molecule on the basis of the spectrum, and these are compared with the conclusions of the problem-solving

module. The system can confirm the student's hypothesis, or indicate why it is wrong (i.e. what contradictory evidence is present). It can also explain why incomplete student statements are defective.

Since the ACE system solves the problems in a similar manner to an expert, it can also give a partial or complete demonstration solution to the student if required.

However, although ACE identifies incorrect steps in a student's arguments, it has no way of representing systematic student errors or misconceptions, and is therefore unable to predict problems which will occasion difficulty in the future, or to classify general topics in need of remedial explanation.

Domain Experts with Student Models

A program that actively teaches a student, rather than one which gives him practice in something he already has a basic competence in, must have some way of inferring the student's knowledge state (Self 1974, Ohlsson 1986). This is necessary for the tutorial program to be able to select appropriate questions to present to the student, and for it to select appropriate remedial actions in the event of the student being unable to solve a problem.

Frame-based programs have no explicit student model (other than perhaps cumulative scores of right and wrong answers), but they can embody considerable tutorial sophistication in the design of their frames and their sequential relations with each other. Any such analysis depends on the subject expert who defines the teaching sequence. Its usefulness is in practice limited by the combinatorial explosion described earlier to dealing with only a handful of the conceivable cross-product of knowledge states and example problems.

Therefore a natural development from generate-and-test systems are domain experts incorporating student models. Such programs

are sometimes referred to as "intelligent" tutorial systems, because they have the potential for adaptive (i.e. "intelligent") interactions with the student.

As an elaboration of generate-and-test systems, intelligent tutorial systems are even more complex, and consequently they are expensive to write in terms of the time needed (at least two orders of magnitude dearer than frame-based programs); and typically they require large systems on which to run. This last feature is becoming of decreasing importance as the cost of computing declines, but the cost of writing the software remains a large overhead.

In order to compare the student's answers with those of the domain expert module, it is of course necessary that the domain expert should solve problems in the way that it is desired that the student should do. A "black box" expert which produces answers by a method having no relation either to the student's actual or target thought-processes cannot be used as the basis of an Intelligent Tutorial System. A degree of psychological validity is required of the problem-solver, especially as regards the generation and sequencing of its high-level goals. If this is not the case, then the task of comparing the student's output with that of the expert module in order to identify the student's errors becomes impossibly difficult.

Types of Student Model

The representation of the student's knowledge, or "student model" can be:

(a) Implicit

(b) Stochastic

(c) Overlay Model

(d) Malrule Model

(e) Student Simulation

Implicit Models

A system can be said to have an implicit student model if it behaves as if there were a model of the student's knowledge embedded in the program, while there is actually no explicit data-structure fulfilling such a role. Such models are common in frame-based programs, but the dissociation of domain knowledge from specific problems inherent in programs with domain experts makes an implicit model unrealistic for more sophisticated systems.

Stochastic Models

A program has a stochastic model if it represents the student's knowledge as a set of conditional probabilities that given particular input states, they will respond with particular outputs. Such a model is only applicable to domains with enumerably many states, such as vocabulary learning.

Overlay Models

An overlay model is one in which various parts of a correct procedural domain knowledge representation are marked as "known" or "unknown" by the student. That is, it assumes the student's knowledge is a subset of the domain expert's.

A correct answer by the student to a problem for which the domain expert uses facts F1, F2, and F3; and rules R5, R6, and R7; would therefore constitute evidence that the student knew F1, F2, F3, R5, R6, and R7.

If the tutorial goal of the system were to ensure that the student knew all relevant facts and rules in the domain, it could select the next problem for the student by choosing a question whose solution would involve facts and rules not known to have been learned by the student. Also, at the end of a tutorial dialogue, the program can in principle inform the student of what elements of the syllabus they are known to have mastered, and which elements they are not.

An example of a tutorial program with an overlay student model is the BIP system (Barr, Beard and Atkinson 1976), which teaches programming in the BASIC language. The program has a "curriculum" of concepts which it is intended that the student should learn, and it selects problems for them to solve in such a way as to check their competence in all topics on the curriculum with a minimum of redundant effort. This is done by maintaining a list of topics which may be used in a question - the MAY set. This comprises all topics at the current level of difficulty in the curriculum or less. It is therefore implicitly assumed that the student will have been taught these topics, and so they may be made use of in questions.

A list of topics on which the student needs further work, on his own estimation or that of BIP, is called the MUST set because they are topics that he MUST learn more about. In general, the MUST set will be a proper subset of the MAY set.

A problem is selected for the student using only skills in the MAY set, which maximises the number of skills required from the MUST set. This ensures that he will be directed to work on topics that need practice, without introducing topics he hasn't yet covered.

The main problem with overlay models is that in general, a student's knowledge will not consist simply of a subset of the expert's knowledge, but will include errors and misconceptions: errors of fact and misconceptions as to how to carry out algorithmic procedures. These cannot be dealt with by an overlay model, and yet from the tutorial point of view, the errors and misconceptions in a student's knowledge are perhaps the most important features of that knowledge - good educational practice demands that they be sniffed out, hunted to earth and destroyed like rabid foxes; and not just ignored in the frivolous hope that they will go away.

Modelling errors of fact is fairly straightforward- if the program contains a list of the capitals of different countries, and the student appears to think that Gorgonzola is the capital of Italy, all that is required as a student model is a list to record that the student thinks Gorgonzola is the capital of Italy.

Representing erroneous procedural beliefs is more challenging. Firstly, the program will probably not have access to intermediate levels of working, but will have to infer the error from its result. Secondly, more than one erroneous procedure may produce the same result. Thirdly, some erroneous procedures may result in there being no output at all. How do

you tell what procedure produced the behaviour when there is no behaviour? This is an example of the so-called "assignment of credit problem" (Sleeman and Brown 1982), and is a major unsolved problem for cognitive science as a whole.

In order to represent erroneous procedures in a student model, it is necessary to have an accurate and largely complete theory of what erroneous procedures are possible in a given domain.

Malrule Models

If a procedure carried out by a human problem-solver is represented by a set of rules in a formal language, then an incorrect procedure can be represented by a procedure containing one or more incorrect rules. These are known as "malrules" (Sleeman 1981). The essential features of a malrule are:

(1) In the context of the correct problem-solving rules, the malrule-containing procedure is capable of generating the incorrect target behaviour.
i.e. Malrules are generative.
It is this feature which distinguishes malrules from other forms of error description, and makes them more precise and more useful.

(2) Malrules are in some sense minimally psychologically representative. That is, they cannot be factored into smaller units which are themselves psychologically accurate, but in themselves are intended to represent psychological processes that take place in human problem-solving.

Thus, a description of a student's problem solution in a chemical domain as "The student is unable to distinguish different halides in ionic solution", is not a malrule, because it does not explain what the student actually does do in a given situation.

A procedure which could account for a wide range of correct and incorrect behaviour, but which had nothing in common with a totally correct procedure would not be considered a malrule either, because it would not be a replacement for a small part of the correct procedure, but rather an alternative theory.

In this work, the word "misconception" is used in a vague way to mean "incorrect fact, algorithm or heuristic"; while "malrule" is used in the more specific sense given above.

Student Simulations

Ohlsson (1986) has proposed a category of student model called "student simulations", which go beyond the inclusion of malrules into a correct procedure. Such a Student Model would also incorporate knowledge of the student's heuristics, goal

structure, and other meta-level control concepts. In principle, such a simulation would be executable. As yet, no published results describe systems which go beyond the "domain expert with malrules" level of student model.

Are Malrule Models Necessary?

An intelligent tutorial system incorporating a malrule model should in principle be able to recognise and note, not only correct student actions, but also incorrect ones. In several domains, the only method yet discovered for representing erroneous behaviour is one based on malrules. At present, therefore, only a malrule-based approach is capable of analysing and diagnosing erroneous student algorithms, and therefore only a malrule-based tutor is capable of correcting them. Although this is a significant potential advance, it is bought at a high cost in programming effort. Not only must a procedure be written that embodies a correct problem-solving procedure, but procedures must be written that embody all, or nearly all, the incorrect methods that occur in real-life students. These may be numerous and obscure and can only be found by experiment. Thus to the task of writing one correct

method is added that of finding N incorrect methods, and then writing the N incorrect methods as formal procedures. It seems that with every step towards more effective and intelligent CAL programs, there is a massive increase in the intellectual overhead involved.

Tait, Hartley and Anderson (1973) claimed that the effectiveness of an instructional program is largely determined by the program's ability to locate the causes of students' errors, and to provide feedback which draws the student's active attention to these causes. Self proposed a teaching program in which "An embodiment of these errors as procedural bugs gives scope for various kinds of feedback processes" (Self 1974). He advocated the development of student models capable of representing erroneous procedures as a key factor in developing more effective CAL programs.

Thirteen years later, few programs have been written using any student modelling system more complex than an overlay model. This must largely be attributed to the size of the task involved in collecting, identifying, and procedurally representing a significant proportion of the errors that occur in any non-trivial domain.

One system that has attempted to build a malrule model of the

student working in its domain is HELPERR (Jones and Tuggle 1979). This system has been developed as a working system to teach addition, subtraction, multiplication and division of integers. The program, written in the FORTRAN language, generates arithmetical problems of the appropriate type for a student to solve at a terminal. The student's answer is analysed, and if it is not correct, the answer is compared to the possible outputs that would be generated by the errors in the program's taxonomy. These are derived from previous non-computational analyses of arithmetical errors in the domain. In the event that a systematic error is diagnosed, the student is immediately informed of the nature of the error, and given a step-by-step demonstration solution of the problem. As questions are generated rather than stored, a student can be given indefinitely many problems to solve. The "student model" here consists of a list of those skills the student has covered, together with a list of the systematic errors (or "malrules" in our terminology), which he has exhibited.

It was hoped by the creators of HELPERR that the error diagnosis would be helpful to the student's teacher on its own, whether the automatic remedial actions of HELPERR were effective or not. Interestingly, Jones and Tuggle reported that the remedial actions of HELPERR were not very effective. It was expected that immediate description to the student of

the nature of his error would in itself be an effective form of treatment, but his expectation was not borne out by experience. Jones and Tuggle report "This did not appear to work very well. Quite probably several factors are involved".

Anderson and his co-workers have developed tutoring systems guided by student models for the domains of LISP tutoring (Farrell, Anderson and Reiser, 1984), and Geometry (Anderson, Boyle and Yost, 1985). In these systems, the domain expertise is represented in the form of production rules, and malrules are represented in the same format. The LISP tutor, for example, contains 325 correct rules, and 475 malrules, which have been extracted from protocols. The student model is an overlay of this collection of correct rules and malrules, and the tutorial goal of the system is to communicate all the correct rules and none of the malrules to the student. The student's internal state is inferred by monitoring each step, and comparing it with the results of every possible applicable rule; a process termed 'model-tracing' by Anderson. Every time the student makes an error, he receives correction based on his inferred malrule or lack of a correct rule in his current situation.

Test on small numbers of students have been claimed to show excellent results, both in terms of objective scores and also of student motivation.

Other researchers have also been convinced that accurate identification of the precise malrules operating within a student's knowledge base are essential to effective tuition. Stevens, Collins and Goldin (1982) analysed protocols of tutors discussing the causes of rainfall with individual students, and concluded that "Our analysis of dialogues shows that tutors spend a good part of their time diagnosing conceptual bugs from errors manifested in the dialogue. We believe that much of the teacher's skill as a debugger depends on knowledge about the types of conceptual bugs students are likely to have, the manifestations of these bugs, and methods for correcting them. At the present time, there does not exist a large enough body of research to answer with any confidence questions about the level of detail of student model needed for effective teaching, either by humans or machines. Such a body of research could only be done if there existed domain expert programs incorporating highly sophisticated student modelling components able to act as a test-bed for different tutorial strategies."

Error Modelling in Problem-Solving Domains

Although few teaching programs have to date incorporated malrule-based student models, there have been a number of systems which attempt to model both correct and incorrect problem-solving behaviour as an end in itself. One domain in which considerable work has been done is that of the subtraction of integers.

Brown and Burton (1978) collected several thousand scripts by school students solving subtraction problems. From these, they analysed about a hundred and ten malrules (which they term "bugs"), each of which leads to specific wrong answers when used with subtraction problems. They wrote a LISP program called DEBUGGY which, when given the answers of a student to a set of problems, diagnoses the state of the student's knowledge in terms of the malrules present. This was done by comparing the answers produced by the student to the answers produced by LISP procedures contained in the program, each of which produced the same answers as one of the malrules.

Because the diagnosis takes place off-line after all the student's solutions have been produced, the program is able to

compare its student model (i.e. set of malrules hypothesized as being present), with the whole of the student's output- not just with one answer at a time. DEBUGGY can not only be used to diagnose individual students, it can also analyse a proposed test in order to see if the questions given would distinguish different possible student malrules.

The DEBUGGY model of subtraction deals only with procedural knowledge as it exists at one moment in time. It embodies no theory of how such knowledge may have developed, (in this it resembles HELPERR). Such a theory is clearly called for, because many of the malrules identified by DEBUGGY correspond to actions which are not taught explicitly in arithmetic classes. Therefore the question arises: where do they come from? This issue was tackled by Brown and Van Lehn (1980), who developed a theory of malrule genesis called "Repair Theory". A later version of this is to be found in Van Lehn, (1981).

The essence of Repair Theory is as follows:

(1) A procedural skill is made up of subskills, which may be learnt independently.

(2) At any time, a student possesses a "core" of correct subskills which have been learnt.

(3) If the student tries a problem which cannot be solved using the correct core procedure in his possession, he will at some stage reach an "impasse", that is, a point at which his core knowledge cannot be applied to proceed with the problem.

(4) When a student reaches an impasse, he applies one of a small number of domain-independent procedures called "Repair Rules", to his incomplete core procedure. The repair rule generates an extension to the procedure which may be correct, but may be a malrule.

(5) The "repaired" procedure is then executed, producing a (possibly incorrect) result.

The initial rationale behind this theory was the insight that a

typical defective computer program exhibits halting behaviour when executed, but human problem solvers often do not. For some reason, people avoid halting behaviour, even when equipped with inadequate problem-solving procedures.

Brown and Van Lehn believe that the particular repair rule selected is independent of the impasse that made it necessary. They further make the strong claim that repair rules are domain-independent. It is not clear how they arrive at this conclusion, since their published work relates exclusively to subtraction, a domain in which the algorithms correspond only indirectly to intuitive concepts.

Van Lehn believed that from the nature of the repairs one could gain insight into the structure of the underlying knowledge representation used by the human problem-solver. He chose to examine two particular operators- the deletion operator that "forgets" core procedures, and a repair rule called "backup", that returns the solver to an earlier state when an impasse is reached.

Alternative underlying control structures were considered which might account for these repairs, and any that were clearly insufficient to produce the relevant behaviour were rejected. Any which required unlimitedly large resources in the way of

memory or other requirements were also rejected. This left only one possible underlying architecture to be considered- a result that did not depend sensitively on estimations of algorithmic efficiency. Van Lehn says, "By studying the kinds of changes they make, their computations, one can understand what requirements must be placed on the mental representations that they manipulate." The control architecture inferred by Van Lehn for a human problem-solver using repair rules possessed the following features:

- it must be possible for the interpreter to return to an earlier execution state, from which an alternative control path to that chosen previously, (this is the constraint that backtracking should be possible).

- when control is returned to an earlier state, the data values are reset to the values that obtained in the earlier state as well.

- the interpreter must be capable of coping with recursive procedure calls.

- the procedure representation language must be hierarchical in that it supports the notion of goals with subgoals. When backtracking occurs, the interpreter searches up the hierarchy starting from the current goal, going up from goal to supergoal, until it reaches one that can return an alternative result. This is equivalent to keeping the procedure goals on a stack.

Van Lehn says " . . . the execution state should be a stack, which entails that the knowledge representation for core procedures has a goal-subgoal calling hierarchy".

- the goals are typed as AND goals or OR goals. An OR goal succeeds if any of its subgoals succeeds, and an AND goal succeeds if all its subgoals succeed. Clearly, when the interpreter starts backtracking, it must backtrack as far as an OR goal.

- the language used is applicative. That is, all data is passed via procedure parameters, and there are no common variables at all. The only other way data can be accessed by different goals is by being stored externally on paper, for example.

Interestingly, all of these features are possessed by the PROLOG language, a fact which Van Lehn does not mention.

Young and O'Shea (Young and O'Shea 1981) examined the same field of subtraction problems, and also produced a computer model to account for the errors made by students. Its coverage of student errors was comparable to that of DEBUGGY, but its representation was different. Instead of LISP procedures, the student's method was represented by a set of production rules (Davis and King 1977). A correct subtraction strategy on the part of the student corresponded to the presence of all the relevant production rules. Erroneous results are obtained by the omission of productions, or the inclusion of incorrect productions- most of which relate to calculations involving the zero digit. Young and O'Shea claim that "...the algorithm errors can without exception be accounted simply by omitting rules from the correct set", and hence the need to repair defective procedures does not feature highly in their work.

A similar approach towards accounting for systematic student errors has been taken by Sleeman and Smith (1981), with respect to the solution of linear algebraic equations. The Leeds Modelling System (LMS) represents the solution of linear algebraic equations such as:

$$2 * X + 4 * X + 4 = 16$$

by production rules which successively simplify the expression until a solution is reached. Only eleven such rules are required, and in order to generate systematically erroneous solutions, some incorrect "malrules" are necessary; one or more of which are added to the rule set. In this domain, Sleeman and Smith were unable to represent erroneous solution paths by simply omitting rules or including rules for other domains. Nor did they hypothesize an independent repair mechanism. The malrules used bore a marked syntactical resemblance to the set of correct rules, but this resemblance has not been formalized in such a way as to enable their generation.

Problem-Solving in Physics

Computational systems able to solve problems in the domain of Physics have been developed by Larkin, McDermott, Simon and Simon (1979, 1980), and by Bundy, Byrd, Luger, Mellish and Palmer (1979). Of the two, the system developed by Larkin et al is more concerned with psychological and educational issues, especially with structural differences between the knowledge of experts and that of novices.

Since these systems relate directly to the topics dealt with by NEWT, they will be studied in more detail than the other systems described, and will be dealt with fully in the next chapter.

Conclusion

This chapter has looked at some of the different types of computer-assisted learning that have evolved over the past twenty years; and has examined the case for basing computerised tutorial systems on an accurate psychological model of the student- including a model of the student's errors and misconceptions.

Some different approaches to modelling student expertise and student errors have been examined, and in particular the malrule approach has been investigated with respect to the domains of integer arithmetic and the solution of linear equations.

In the next chapter, work on the computational modelling of physics problem-solving will be looked at, and compared to existing experimental data. Conclusions about the problem-solving structures used by experts and novices will be inferred from this analysis, which have implications for the construction of tutorial programs in this area.

Chapter IV Control Strategies and Problem Representation in Physics Problem-Solving

This Chapter examines published studies of human problem-solving in the domain of Newtonian Mechanics; with particular reference to theories of control strategy and problem representation.

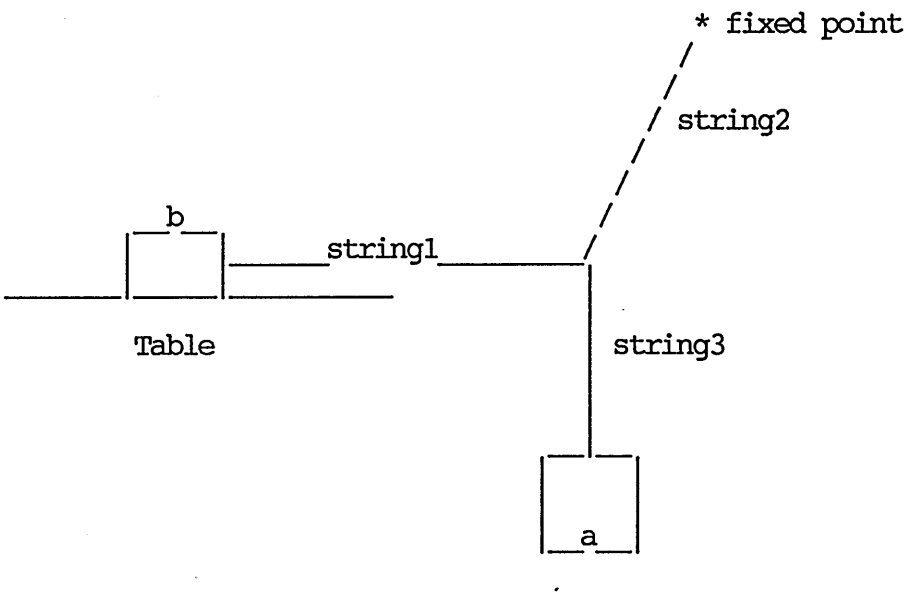
Existing theories in this area are compared with the data available, leading to the formulation of a control strategy termed "Planstacking", and to two mechanisms for knowledge representation termed "Sketch Construction" and the "Hidden Curriculum Assumption of Non-Redundancy".

The Problem Domain

There is a wide range of problems in Physics, whose solution may be described like this:

Problem Statement	-->	Generate Simultaneous Equations	-->	Solve Equations	-->	Answer

Domains for which this applies include kinematics, dynamics, statics, thermodynamics and DC circuit theory. Here is an example:



Problem: A block of mass mb rests on a rough horizontal plane, whose coefficient of friction is mu, the block being in limiting equilibrium. The block is pulled by a horizontal string attached to two other strings, one at an angle of 45 degrees to the horizontal whose other end is fixed, and one hanging vertically. On the other end of the vertical string a mass of ma is suspended. Find ma.

As this particular example has been studied by a number of researchers, I shall use it here as a "test-bed", to demonstrate various possible solution methods.

All solutions require a number of applications of the "resolution of forces" principle applied in a number of different situations. Each application of the principle generates an equation, and the set of equations generated allow one to solve for the sought unknown. We shall not dwell on the question of how the problem is translated from natural language, or how the equations are solved, but will focus on the intermediate "Physics" stage of the solution.

Three possible control strategies for generating equations are described: backward inference, forward inference, and backward planning followed by forward equation generation (referred to here as "planstacking"). These are the strategies that have been proposed by various workers as descriptions of human problem-solving methods.

Backward Inference

When using this method, the problem-solver starts with the sought unknown, and generates an equation which contains that unknown (and preferably no other unknown). If there is still another unknown left to find, this procedure is repeated until an equation has been generated for every unknown.

If the tensions in the three strings, string1, string2, and string3 are taken as t1, t2, and t3 respectively; and the reaction of the table on object b is called "R", then this leads to the following solution:

Principle	Context	Equation	Remaining Unknowns
Resolve Forces	vertically at a	t3 - ma * g = 0	t3
Resolve Forces	vertically at thejunction	t2 * sin 45 - t3 = 0	t2
Resolve Forces	horizontally at the junction	t2 * cos 45 - t1 = 0	t1
Resolve	horizontally at b	t1 - mu * R = 0	R
Resolve	vertically at b	R - mb * g = 0	nil

Forward Inference

With this method, equations are generated if they allow for the evaluation of an unknown quantity. This process is repeated in the hope that eventually the sought unknown will be generated.

Principle	Context	Equation	Known Quantities
Resolve Forces	Vertically at b	R - mb * g = 0	mb, R
Resolve Forces	Horizontally at b	t1 - mu * R = 0	mb, mu, R, t1
Resolve Forces	Horizontally at the junction	t2 * cos 45 - t1 =0	mb, mu, R, t1, t2
Resolve Forces	Vertically at the junction	t2 * sin 45 - t3 =0	mb, mu, R, t1, t2, t3
Resolve	Vertically at a	t3 - ma * g = 0	mb, mu, R, t1, t2, t3, ma

Planstacking

With this method, a principle is chosen which would generate an equation able to solve for the currently sought quantity. Instead of being used to generate the equation immediately, the principle and its relevant context (which will be called a "plan") are stored on a stack. If the equation that the plan would generate would include another unknown quantity, a plan is developed to find this as well.

Successive plans are pushed on the stack until no sought unknowns remain. Then the plans are successively unstacked, and as each plan is popped off the stack, it is used to generate an equation.

Planstack	Soughts Remaining	Equations Generated
Resolve forces vertically at a	t3	-
Resolve forces vertically at junction Resolve forces vertically at a	t2	-
Resolve forces horizontally at junction Resolve forces vertically at junction Resolve forces vertically at a	t1	-
Resolve forces horizontally at b Resolve forces horizontally at junction Resolve forces vertically at junction Resolve forces vertically at a	R	-
Resolve forces vertically at b Resolve forces horizontally at b Resolve forces horizontally at junction Resolve forces vertically at junction Resolve forces vertically at a	-	-

No remaining sought unknowns - now pop stack

Planstack	Equations Generated
Resolve forces vertically at b Resolve forces horizontally at b Resolve forces horizontally at junction Resolve forces vertically at junction Resolve forces vertically at a	R - mb * g = 0
Resolve forces horizontally at b Resolve forces horizontally at junction Resolve forces vertically at junction Resolve forces vertically at a	t1 - mu * R = 0
Resolve forces horizontally at junction Resolve forces vertically at junction Resolve forces vertically at a	t2 * cos45 - t1 = 0
Resolve forces vertically at junction Resolve forces vertically at a	t2 * sin45 - t3 = 0
Resolve forces vertically at a	t3 - ma * g = 0

Comparison of the three methods

Of course these three problem-solving control strategies are not the only ones possible. It is quite possible for someone to generate the relevant equations in any order. In particular, they could generate one or two equations by forward inference, and then finish the problem by backward inference, or they could start by backward inference and then finish by forward inference, or they could build a planstack and pop it before all unknown quantities were accounted for. However, we shall restrict this analysis to these three well-defined strategies.

Backward Inference

This method guarantees that only relevant equations are generated. Any new unknown introduced can be identified by a simple syntactic check of the equation generated. It is not possible to solve any of the equations until the last one has been found, because only the last one will contain a single unknown. It is similarly impracticable to collapse equations together by substituting expressions for one unknown into a subsequent equation.

Forward Inference

Irrelevant equations may be generated if irrelevant information is present. Note that the given problem, like many textbook problems, does not contain irrelevant information, so that this question does not arise here. New unknown quantities can be identified easily. Each equation can be solved as it is generated, and the value of the corresponding quantity substituted into the next equation when it is generated. This makes the equation solution a much easier process than for the backward inference method. Notice that on the example given above, forward inference generates equations in exactly the opposite order from backward inference- an observation which holds for many problems in this domain.

Planstacking

This will not generate irrelevant equations, and each equation can be solved as it is generated. Equations are generated as the stack is popped in the same order as by forward inference. It is necessary for the problem-solver to be able to infer what quantities would be present in an equation without actually generating the equation, and it is also necessary for the problem-solver to have enough working memory to hold the planstack.

General Comments

I have worked through the implications of these three control strategies in some detail, because knowledge of the problem-solver's methods can most reliably be inferred from their actions. It is therefore necessary to be familiar with the consequences of all possible hypotheses in order to know what a given observation implies. Larkin (1979), Luger (1977, 1981) and Van Lehn (1981) all make claims as to the nature of the human problem-solving processor, based on inferences from subjects' solution traces. As we shall see, these claims are incompatible with each other, and can only be evaluated by considering the consequences of the different problem-solving strategies that these researchers use in their respective models.

The MECHO program

The MECHO program was developed by Bundy, Byrd, Luger, Mellish and Palmer (1979). It was designed to take a statement of a Physics problem and generate from it a set of equations that would solve for the sought unknown or unknowns in the problem. In concert with this, the MECHO team developed a natural language interface able to translate problems in English into predicate calculus assertions which the problem-solver uses as its database. They also developed an equation-solving program known as PRESS, which was able to solve the sets of equations that MECHO generated.

MECHO was written in PROLOG, and was intended as an efficient domain expert rather than as a psychological model. However, the basic control strategy derives from the work of Marples (1974), who devised the "Marples Algorithm" as a teaching aid for undergraduate Engineering students, and so is based on an analysis of human performance. The Marples Algorithm is a specialised form of "means-ends" analysis- that is, it works by backward inference.

Firstly, a plan is selected which will generate an equation containing the sought unknown, and if possible, no other

unknown quantities. The plan consists of a principle, together with a context in which the principle can be applied. When the plan has been created, it is used to generate an equation involving physical quantities. This equation is then checked to see if it includes the sought unknown. If it does not, MECHO backtracks and creates an alternative plan. If the equation contains only the sought unknown, MECHO has finished. If it contains the sought unknown and another unknown, MECHO marks this new unknown as "sought", marks the original sought as "known", and the Marples Algorithm is called recursively in order to solve for the new quantity. Thus MECHO contrives to generate new equations until no sought unknowns remain.

Thus the algorithm depends crucially on three things:

- selecting a principle;

- finding a context in which to apply the principle;

- applying the principle in the context in order to generate an equation.

Selecting a Principle

Everything dealt with by MECHO has a type. Thus, all forces have type "force", all velocities have type "velocity", and so on. The type of a quantity can be deduced from its original definition- that is, from the original assertions in the problem statement that introduced the quantity. MECHO also knows which types of quantity are related together by each principle. Thus the "conservation of momentum" principle relates together quantities of types "mass" and "velocity", while the "taking moments" principle relates together quantities of types "force" and "distance". Since this is knowledge about principles rather than objects, it is termed "meta-level knowledge" by the MECHO team, as opposed to "object-level knowledge".

MECHO uses its meta-level knowledge to create a shortlist of principles that involve quantities of the same type as the sought unknown. The principles on the shortlist are used in succession in attempts to form equations containing the sought unknown.

Finding a Context

In the "test bed" problem, MECHO would first note the type of "ma", the sought unknown. This quantity has the type "mass", and the meta-level knowledge about principles allows MECHO to shortlist "Resolve Forces", and "Take Moments", as possible principles to use. Each principle requires a different kind of context in which to be applied.

"Take Moments" needs to be applied to a rigid body about a point. "Resolve Forces" needs to be applied to a physical object in a particular direction. Again, the type information about the objects is used to find an applicable context. The only rigid object mentioned in the question is the table. This is not in contact with an object having mass ma, and so the "Take Moments" principle cannot be used.

"Resolve Forces" can be applied to body "a" because it is a physical object.

Thus meta-level reasoning about the types of objects in the problem, and the contexts required for the application of particular principles, are used to produce a plan.

Applying the Plan to Generate an Equation

There are different procedures for generating equations from different principles. In the case of "Resolve Forces", a list is made of everything connected to the object being resolved for, and then the force due to each is calculated. To these are added the weight (if any) of the body, its reaction on the surface on which it lies (if any), and the frictional force (if it exists). The sum of these forces is equated to zero when the object is known to be stationary, and this total is the desired equation.

Database Inference

When MECHO accesses its database of facts about the problem, it needs to recognise things which a person would take to be "obvious", but which are not explicitly stated in the problem, and which do not correspond to the application of physical principles to generate equations.

Examples of such knowledge might be:

- recognising that if ABC is a straight line, and AC and AB are given, then BC = AC - AB.

- recognising that the tension in a string is the same on each side of a light frictionless pulley.

- recognising that if two particles are in fixed contact over a period of time, their relative accelerations and velocities are zero throughout that period.

MECHO has two ways of generating such information:

(a) When the problem is first read in, a number of special-purpose schemata recognise particular object configurations, and create database assertions to fill in some "obvious" facts about them. For instance, one schema asserts that if a string passes over a light frictionless pulley, the tensions are the same on each side of the pulley.

(b) When a particular quantity is required by the equation-generating procedure, it is not simply looked for in the database, but accessed by a special-purpose database inference mechanism. This mechanism performs "obvious" inferences, such as the first of the examples given, and is also responsible for creating new names for quantities it cannot evaluate.

The choice as to which "obvious" inferences are handled by which mechanisms has largely been one of practical convenience- such inferences as seemed certain to be necessary were handled by schemata, while others were left to the database handler.

MECHO and Human Problem-Solving

MECHO was designed as a system which would implement a number of computational techniques in a complex problem-solving domain. These techniques included the control of search by meta-level inference, the use of object-related schemata to provide default values for quantities not explicitly mentioned in the problem, and a number of domain independent but special

purpose mechanisms for reducing search when accessing the database
(Bundy, Byrd and Mellish 1982). It was not intended mainly as a psychological model.

However, since the central solution strategy was derived from an analysis of human problem-solving, MECHO has been compared by Luger with human problem-solvers in an attempt to test the similarity between MECHO's solutions and those of experimental subjects (1977, 1981).

Luger found that subjects generated the same equations as MECHO, and introduced names for the same intermediate values as MECHO. They picked the same idealisations of physical objects as MECHO, and selected the same default values. Thus, falling objects were idealised as possessing no air resistance, beams were idealised as being infinitely stiff under bending moments, and so on. As Luger remarks:

> "These necessary assumptions are not stated in the problem, but must be understood for a successful solution."

To an expert, these idealisations come without conscious effort, but they are almost never explicitly stated in the

problem. Default values as used by MECHO are similar to idealisations, except that they deal with object knowledge which might contingently be otherwise; and so default values are not used unless no alternative has been explicitly stated in the problem. Examples are: the default properties of a string being light and non-elastic, and the default values of a surface being that it is straight and has no friction.

Sometimes Luger's subjects generated equations in the same order as MECHO, and sometimes not, but Luger did not relate the order of equation generation to any other particular factor.

Luger's study suggested a considerable degree of congruence between MECHO and human problem-solvers at the level of selecting physical abstractions and generating equations.

It might naively be thought that such questions as "How do people decide which equations to write," were open to introspective analysis- but we shall see presently that completely incompatible interpretations of human problem-solving are possible, and therefore such questions must be investigated more formally.

In addition to Luger's claims for the psychological validity of the Marples Algorithm, and the physical object schemata, we should consider the plausibility of the separation of the system's knowledge into object- and meta-rules. This separation permits knowledge to be held in modular units, each with a declarative interpretation as well as a procedural one. It is hard to see how the psychological accuracy of such a separation could be tested rigorously, but the motivation for such an assumption is supplied by the facts that:

(1) People learn skills gradually- it is hard to see how this can be done without a modular representation of procedural knowledge.

(2) People can generalise problem-solving skills to related domains. It is hard to see how this could happen unless much of their problem-solving knowledge were domain-independent. The separation of object-knowledge from meta-knowledge expressed in MECHO makes it possible to consider modifying part of the problem-solver in isolation, and therefore such a knowledge representation would be appropriate to a system that could generalise its skills.

As Bundy says ".....controlling search by using meta-level inference is superior to built-in smart search strategies because the search information is more modular and transparent. The argument is for systems to make explicit the full knowledge involved in their behaviour, which in turn aids the modification of their data and strategies, thus improving their robustness and generality. This leads the way to systems which could automatically modify their strategies and explain their control decisions." (Bundy et al 1979)

Chi, Feltovich and Glaser

These authors conducted a study on the methods used by experts and novices to solve physics problems (Chi, Feltovich and Glaser 1981). They did not construct a computer model, but relied on subject protocols, solution traces, and an exercise in which experts and novices were asked to categorize problems without solving them.

Their elegantly planned experiments were able to complement each other by approaching expert behaviour from different perspectives. The experimental data may be summarised thus:

(1) Experts took longer to classify problems than novices.

(2) Experts often tentatively hypothesized the principle that would solve the problem before reading the whole question.

(3) Experts classified problems by the principle that would be used to solve them, and novices classified problems by the surface features mentioned in the question. When experts were given questions which referred to the same objects, but required a different sought unknown, and were therefore solved by a different principle, they classified them by the different principles required.

(4) There were signs of intermediate stages of expertise between novices and experts- the distinction was not all or nothing.

(5) When asked to explain how they had solved a problem, experts talked about principles, and novices talked about the knowns and givens in the problem.

The authors explained these findings by proposing a noncomputational model of the process by which both experts and novices solve problems. They summarized their model thus:

" . . . experts engage in qualitative analysis of the problem prior to working with the appropriate equations. We speculate that this method of solution for the experts occurs because the early phase of problem-solving (the qualitative analysis) involves the activation and confirmation of an appropriate principle-oriented knowledge structure, a schema. The initial activation of this schema can occur as a data-driven response to some fragmentary cue in the problem. Once activated, the schema itself specifies further (schema-driven) tests for its appropriateness. When the schema is confirmed, that is, the expert has decided that a particular principle is appropriate, the knowledge contained in the schema provides the general form that specific equations to be used in the solution will take."

It was thought that novices lacked such powerful data-driven schemas, and therefore had to fall back on weak general methods such as backward inference. Experts' schemata were thought to contain applicability conditions in terms of high-level features, while those of novices were in terms of lower-level features. High-level features are relations between the

physics quantities involved, and low-level features are defined in terms of physical objects.

This work extends the findings of Hinsley, Hayes and Simon (1977) from the domain of algebra word problems to that of Newtonian Mechanics. Hinsley et al identified eighteen categories of problem, one of which was "Physics". The problem categories found by Chi et al correspond to sub-categories of this single category- it is not clear whether they regard this as a different level of categorisation, or a finer grain of categorisation at the same level.

Some of the examples given by Hinsley et al are hard to imagine being categorised by any process other than schema recognition. For example, some subjects categorised a problem after hearing only the words "A river steamer..." The problem was categorised as a "River Current" problem. One subject said "Its going to be one of those river things with upstream, downstream, and still water. You are going to compare times upstream and downstream- or if the time is constant, it will be the distance". Since no physical quantities were mentioned, this does not appear to be a classification based on an inference about quantities or physical principles.

Possible interpretations of these results as applied to the Physics domain might be:

-Hinsley, Hayes and Simon's subjects corresponded to Chi, Feltovich and Glaser's novices, who classified problems by the words that appeared in them.

-There are two or more levels of problem categorisation: the more general levels being done by cueing schemata or keyword recognition, and the more detailed levels studied by Chi et al being performed by other mechanisms.

-Some objects appear only in problems of a certain type, and hence cue schemata based on keyword recognition. Other objects appear in many types of problem, and do not do so. This would account for the large difference in the number of words needed to recognise problems of different types. Hinsley et al found that on average only three words were needed to recognise a "scale conversion" or a "river current" problem, but seventeen were needed to recognise a "work area" problem, and fifteen to recognise a "progessions area" problem.

The Model as an Explanation of the Data

Experimental findings (1) and (2):
This model succeeds well in explaining the two curious findings that experts can guess the principle they will use to solve the problems, and yet take longer than novices to categorise problems.

As the problem is read, fragmentary information cues a relevant schema, which enables the expert to make a plausible guess as to which principle will prove relevant. He does not commit himself to this, however, until all the relevant applicability tests contained in that schema have been performed- which will not be until the whole problem has been read in. The two findings about the timing of problem solution provide an extremely narrow gap for any theory to squeeze through.

They both appear to rule out straightforward backward inference on the MECHO basis as the solution method used by experts. This happens firstly, because a backward inference problem-solver will not know which principles it is likely to use until it has read in the nature of the sought unknown (which typically appears at the end of the problem). Secondly, such a problem-solver is unable to identify any principles it will

use after the first one, without actually generating equations. Therefore it could not categorize problems it had not solved except in the case that they could be solved by a single equation. The first objection, but not the second, applies to a problem-solver based on a planstacking control regime. However, it would be perfectly possible for a problem-solver to use schemata for the purpose of producing a shortlist of possible principles to use, and then to plan the problem solution by a planstack. The initial shortlisting would serve to constrain search during the secondary planning phase. This would have the advantage that the schemata used could be fallible (they could propose principles that were later rejected as unhelpful), whilst Chi, Feltovich and Glaser propose a model of expert behaviour that becomes untenable if infallible schemata cannot be found (because they call for equations to be generated as soon as a principle has been selected). On the other hand, if infallible schemata CAN be found, the planstacking stage would seem to be superfluous.

Can these contending strategies be tested experimentally? The authors imply that the schema-cued conditions of applicability are tested while the question is being read. A protocol from the paper by "expert J.L.", appears to show a principle being decided on at the same time that the problem is read in. Therefore one would assume that problem

categorization will be substantially complete when the problem has been read in, and that experts take longer than novices to categorize problems because they take longer to read them.

If, however, an expert uses only simple syntactic clues to suggest possible principles, but follows the reading of the problem by creating a planstack, he would have a significant processing task to complete after reading the question, and one would expect the expert to read the problem quickly, and then pause for a certain time before categorizing the problem.

The timing data published by Chi, Feltovich and Glaser unfortunately do not allow this sort of analysis to be made.

Priest and Lindsay (1986) carried out an experimental study to compare the theories of Planstacking and Schema-Guided Forward Inference. Their results are described in Chapter 8.

Experimental finding (3):

The model explains why novices classify problems by their surface features- they have no high-level schema conditions to work with. Experts have, and classify their problems accordingly. Thus the model successfully explains the data presented. However, it has implications beyond those mentioned above which need to be examined.

Firstly- the novices described in this study are all able to solve most of the problems they are given. This means that they are able to apply physical principles correctly in order to produce equations- and therefore that they must be able to recognise the contexts in which principles can be applied. The authors give no explanation as to why novices are unable to recognise contexts in which principles can be applied when reading or categorising the problem, yet are able to do so when solving it. It is suggested that novices classify problems in terms of objects and forces explicitly mentioned in the question, and that they move straight from this data to generating equations. Yet no evidence is given that it is possible to select principles to apply without first constructing a problem representation of forces, accelerations and so on.

This could be done if one were to suppose the novices had a schema for every configuration of objects- but that would imply that a novice could not solve a problem that was even slightly different from one he had tackled previously- clearly, such is not the case. The onus must thus rest on Chi, Feltovich and Glaser to show that there is some way of using physical principles without having a high-level description of the context in which they can be applied.

Secondly- it is stated that experts classify problems by the principles which will be used to solve them.

A very interesting protocol by "expert J.L." is included, in which she reads and discusses the following problem:

> "A block of mass M is dropped from a height X onto a spring of force constant K. Neglecting friction, what is the maximum distance the spring will be compressed?"

When reading this problem, J.L. guesses after reading as far as "X", that the principle "conservation of energy" will be used- and in fact it is. This is interpreted by the authors as the data "mass in falling from height X", cueing the "conservation of energy" schema, whose application conditions are subsequently fulfilled.

How did J.L. know after the first phrase that "conservation of energy" would be the right principle? It is easy to believe that such a schema might be cued, but then so might others. Here are some of the principles that might be relevant to a problem involving a mass in falling a distance X :-

(i) Conservation of Momentum

(ii) Elastic Rebound

(iii) $v = u + a * t$)
(iv) $s = \frac{(u + v) * t}{2}$)
(v) $s = u*t + (1/2)*a*t^2$) the constant
(vi) $s = v*t - (1/2)*a*t^2$) acceleration
(vii) $v^2 = u^2 + 2 * a * s$) formulae

(viii) Conservation of Energy

How did J.L. know which would be the right one? Unless the protocol is atypical, there must be some reason why principles (i) - (vii) are not likely candidates.

Since the principle was guessed before the sought unknown had been read in, no form of backward inference could have been responsible. An alternative possibility is that J.L. was using a (possibly subconscious) "hidden curriculum assumption" to eliminate the first seven candidates. A possible candidate for such a Hidden Curriculum Assumption might be:

"Irrelevant quantities are not mentioned in questions."

Since the height X is mentioned explicitly, this constraint is sufficient to eliminate principles (i) ..(vii), leaving Conservation of Energy as the only principle that mentions both known quantities. This gives this principle priority as the next principle to be considered.

Hidden curriculum assumptions are not explicitly taught, and one could easily imagine a question which violated them- but it would look rather a peculiar question, like this one:

"A block of mass M is dropped from a height X onto a spring of force constant K. Neglecting friction, how long will it take before the block touches the spring?"

If the experts were using a hidden curriculum assumption to prune forward search, it would explain why novices do not classify problems by solution method- they have learnt the applicability conditions of physical principles, but not as yet the "hidden curriculum assumptions" for the domain which determine what principle will be the most useful to apply.

Another possibility is that experts are capable of the qualitative meta-level inference necessary to infer what quantity could be found from an application of which principle, and a novice cannot do this without first using the principle to generate an equation.

Interestingly, the authors (Chi, Feltovich and Glaser), examined what happened when people were asked to classify problems that looked alike, but were solved by different principles (in some cases, different questions differed only as to the sought unknown).

In all such cases, experts correctly identified the principle which would solve the problem, while novices classified superficially similar problems as alike. In a case like this, schemata which only checked for the applicability of physical principles would duplicate the novice behaviour rather than the expert behaviour. A schema that distinguished problems differing only as to the sought unknown would need to be a very different sort of vegetable- but what? The authors are remarkably coy as to the way in which the sought unknown affects the solution- they do not say whether it is relevant to the schema structure or not. As they put it:

> "Although the problem unknown obviously cannot be ignored by the experts, the status of the unknown in the expert solution method appears secondary to that of deciding which physics principles have their condition of applicability met in the problem."

Yet what are the implications of expert's abilities to categorise differently problems to which the same principles could apply? Do we assume that there is not only one schema per principle, but one schema per possible unknown per principle? This would make rather a large number of schemata necessary. Or do we assume that each schema itself can check

the status of the unknowns it will contain, and use this knowledge in deciding its own applicability?

Either of these possibilities- multiple schemata or schemata that check the status of relevant quantities - would be adequate to explain the selection of the correct principle in problems requiring only one application of a principle to produce a solution. However, in multi-stage problems like the earlier "testbed" problem, data-driven schemas could not know the status of some of the quantities involved, as they are not explicitly defined in the problem. Thus reliance on data-driven schemata to select solution principles involves schemas capable of making use of quantity status when available, but also of selecting the right principle in the absence of such data. How a schema which needs to know the status of the relevant quantities in order to select the correct principle can function without this data is not clear. Since the authors have not subjected themselves to the discipline of producing a working program, they have not had to deal with the question.

An alternative explanation for the ability of experts to choose the correct solution strategy would be that they create a planstack of principles and contexts, guided by backwards inference at the meta-level (Priest 1986). Novices might be assumed to be unable to do this, either because they could not

infer what quantities a given plan would solve for, or because they could not maintain a whole planstack in short term memory, or because their lack of hidden curriculum assumptions led to an excessively large search space when constructing a plan.

Experimental Finding (4): There were intermediate stages of expertise between expert and novice level. This clearly suggests that whatever knowledge experts possess and novices don't is modular. Therefore, it is possible to learn some of it but not all. Since the structure of schemata is not described in great detail, the model is consistent with either a modular or a non-modular implementation.

This data is therefore neutral as regards the model (but would be relevant to testing any implementation of it).

Experimental Finding (5): When asked to explain how they solved a problem, experts talked about principles, and novices talked about the known and given quantities in the problem. This was interpreted as showing that experts concentrated on the applicability conditions of schemata, while novices concentrated on the status of physical quantities.

On examination, some of the "key features" cited by experts do indeed look like applicability conditions- for example, the paper quotes "speed-distance relation", and "inelastic collision", as examples. These would appear to resemble possible high-level schema conditions. Others, however, are not so clear. "Force too complicated", for example, does not sound much like the conditions of a prestored schema- more like the conclusion of someone who has tried to solve a problem by using Resolution of Forces and failed. If a schema is truly data-driven, then trying to solve the problem in order to classify it, in order to solve it; should not be allowed. Of course a schema could be made arbitrarily powerful by including a "do-it box"- a subprocedure which itself solved the problem- but then there would be no point in having a schema in the first place. Another "key feature" cited is, "Don't need details of motion", which is again a reasonable description of a solution after it has been completed- but is this a data-driven problem description? The features mentioned in the paper are not explicitly related to named schemas, and while some resemble "high-level slots", of a prestored schema, others appear more like descriptions of a solution that has been already completed- or at least planned.

Conclusion to the Evaluation of this Model

Chi, Feltovich and Glaser have proposed a non-computational model of physics problem-solving. It has been tested against a variety of experimental data.

It is good at:

- Explaining why experts take longer to categorise problems than novices.
- Explaining why experts can guess the relevant physics principle before reading all of the question.
- Explaining why experts described their problem solutions in terms of applying physical principles.

It is less good at explaining:

- Why experts did not consider irrelevant principles in the solution of a problem.
- How novices were able to apply principles to solve problems they did not categorize by solution principle.
- How experts used the status ("sought" or "given") of unknown quantities to decide between different applicable principles.
- How experts correctly choose principles to solve multi-stage problems, in cases where more than one principle could be applied to a situation in which "soughts", and

"givens" are not explicitly stated in the problem.

Larkin, McDermott, Simon and Simon

J. Larkin has written a number of papers on the solution of physics problems by experts and novices, some by herself alone, some in collaboration with others (Larkin 1979; Larkin 1980; Larkin, McDermott, Simon and Simon 1979, 1980; Larkin 1977; Reif, Larkin and Brackett 1976; McDermott and Larkin 1978; Larkin and Reif 1979). I shall deal with these as a whole, since a strong thread of continuity runs throughout all of them.

Unlike the previous authors, Larkin has been concerned to test her theoretical ideas on working programs. This useful discipline supplies a sufficiency argument to her work- the program is sufficient to produce the behaviour it produces, and therefore the theory is sufficient to produce this behaviour as well, inasmuch as the program is an implementation of the theory. In particular, a working program cannot include a "do-it box", without the fact being clear from any reasonably detailed program description. This is a great advance; because the use of "do-it boxes" is both extremely tempting and very

hard to prove when constructing non-computational theories. This discipline is bought at a heavy cost, however. Not only is it a great deal of work to produce working programs which model psychological theories; but their use introduces methodological problems of its own. These will be considered in more detail later on.

Larkin has produced a number of models, all of which were written in a production system language called OPS2. This was chosen because Larkin and her collaborators believed that human problem-solvers conformed with three constraints:

- they possess a limited short-term memory

- their knowledge, both procedural and declarative, is modular in structure

- long-term knowledge is activated only when it is cued by the contents of short-term memory.

Since these constraints also apply to production systems, Larkin has used them extensively for modelling human behaviour. As she herself puts it:

" . . . . they are fairly transparent representations of verbal theories." (Larkin 1979)

The various computational models have been compared with a number of novice and expert protocols. Such factors as order of principle selection, and the timing of equations as they are produced, were used to test the models.

All the models constructed attempt to mirror the assumed structure of human memory in that they have components representing:

short-term memory - this is the working memory of the OPS2 production system

long-term memory - this is the set of productions themselves

external "paper" memory - this consists of a sequential list of assertions made by productions during the execution of the model.

In practice, both the "short-term memory", and the "paper memory", are segments of computer memory, but they are accessed differently. The "paper memory" is only accessible if a production specifically reads some of its contents into working memory. The contents of working memory are assumed to be available at all times to all productions.

In "Models of Competence in Solving Physics Problems", (Larkin, McDermott, Simon and Simon 1980), the authors consider the different strategies of forward and backward inference. A group of novices and a group of experts solved various physics problems, and the order in which they used physical principles was noted. The authors constructed a problem-solver that worked by forward inference, which had a 94% correlation with the expert protocols in terms of order of principle generation.

They also produced a problem-solving model that worked by backward inference, which had an 89% correlation with the novice protocols in terms of order of principle generation. The conclusion drawn from this was that novices solved problems by backward inference, and experts solved it by forward inference. They also noted that novices mentioned explicit formulae (i.e. - principles that look like equations, such as

v * v = u * u + 2 * a * s in dynamics), and then substituted quantities into the relevant slots in the formula. The resulting equations were then written down, and no attempt was made to solve them until the final one had been generated. As noted earlier, this is usually inevitable with a backward inference solution method.

In contrast, experts collapsed the stages of formula selection and equation generation into one, and frequently did not even write out the generated equation explicitly, but substituted previously found quantities into it as it was written down. Sometimes, the equation was even solved before being written down.

Since the order of principle generation of a planstacking problem-solver will be the same as that of a forward inference problem-solver, this evidence does not distinguish between the two.

Structure of the Models

The two models will be referred to here as the "novice", and the "expert" models. Both models read in the problem from a syntactically formalised problem representation, omitting any natural language analysis.

The models start by assigning an appropriate symbol to each description of a known or desired quantity in the problem. The symbol assigned needs to be appropriate, because it will be used to match a symbol appearing in a relevant physical formula. As the authors put it: "For example, given a time interval and an object's velocity at the beginning of that interval, a human assigns to it a symbol like Vo."

The next stage is to select a formula to use to develop an equation. At this point the two models differ.

The Novice Model

This uses backward inference, beginning with the sought quantity and looking for equations that contain the quantity. When one has been found, it is placed in the "paper memory", and any unknown quantities in the equation are marked as "sought". Then the process is repeated to find equations that solve for the new sought unknowns. The model will give priority to equations which involve only one new unknown. An equation can only be applied in a particular context. For example, the equation:

$$v = u + a * t$$

can only be applied to a time interval- so its context will be a time interval.

The Expert Model

This uses forward inference, starting with the given quantities in the problem, and generating equations from them that contain at most one unknown. These are placed in the paper memory until all quantities have been solved for.

As far as principle selection is concerned, the authors state:

> "Its principle-selection mechanism is merely to note what values for variables are known, and to select a principle which allows the finding of a new related variable."

In addition, the equations formed are automatically solved as they are generated. As we saw earlier, this is often possible with a set of equations generated by forward inference.

The principle-selection mechanism described above leads to irrelevant equations being formed unless it is constrained in some way. The method of constraint chosen is to divide up the available principles into groups called "methods". Examples of "methods" include kinematics, force equations, energy

equations. The first equation is chosen without constraint. Subsequent equations are constrained to be chosen from the same method as the first one unless a point is reached where no further equation can be generated from that method.

For most of the problems investigated, only one context was involved. That is, there was only one object and one time interval to which principles could be applied. The model was then developed to handle multiple contexts by:

(1) Giving priority to equations relating to the same context as the previous equation. This focussed the model's attention on one context at a time.

(2) Introducing meta-level information relating particular types of context. Specifically, they include the knowledge that if two objects have the same position at both the start and the end of an interval, they must have travelled the same distance, and so the quantities representing the distances could be equated. This was used to solve a problem about the motion of two cars.

The Model as Explanation of the Data

In the paper referred to, the data consisted entirely of the sequence of physical principles needed to solve various problems, nearly all of which involved a single context. The model was outstandingly good at duplicating this data.

Since this psychological theory is formulated as a working program, it is clear that it is a sufficient determinant of the behaviour it generated. There can be no reasonable suspicion that any "do-it box" is lurking in the nether regions of the theory. Also, it is successful in explaining the data with which it is associated. There are, however, a number of questions which the description raises which remain open.

These may be summarised as:

(1) Explanatory power as regards other data

(2) Implications of Symbol Assignment

(3) Implications of grouping principles in "Methods"

(4) Semantic ambiguity of "Contexts"

(5) Ambiguous treatment of Variadic Formulae

(6) Possible Combinatorial Explosion of Context Relation Knowledge

1. Other Data

The expert model does not produce the behaviour found by Chi, Feltovich and Glaser as regards:

(a) Ability to guess the principle by which a problem will be solved before its completely read in.

(b) Ability to categorise a problem correctly without solving it.

Conceivably, it would be possible to modify the model to cover behaviour (a). Duplicating behaviour (b), however, would be more difficult because of the possibility of "garden path" problems.

Chi et al. propose infallible schemata to produce this data, but Larkin et al. have had to restrict themselves to methods they could implement. On simple one-context problems, it might be expected that forward inference would usually identify the ideal solution principle- but with multi-context problems, the lack of ability of the forward inference mechanism to look ahead renders it fallible. Consider, for example, the question solved by "expert J.L." for Chi, Feltovich and Glaser.

Were the "expert" model to attempt this, it would have no way of knowing that the "Conservation of Energy" principle was more efficient than one of the kinematic formulae. Worse, if it had picked a kinematic formula, it would have committed itself to that method until an impasse had been reached. The same reasoning applies in principle to many other multi-stage problems.

Chi, Feltovich and Glaser clearly demonstrated the ability of experts to choose the correct solution principle, even in deliberately confusing and misleading problems. In the form described, the "expert model" of Larkin, McDermott, Simon and Simon, is incapable of producing such behaviour with any degree of reliability.

2. Assigning Symbols

The symbols assigned to various quantities at the start of the model are extremely important. MECHO can assign names to quantities when necessary, but they have no control significance. In the novice and expert models, however, quantity names serve as both types and tokens- that is, they indicate the type of a quantity (time interval, mass, initial velocity, etc.), and also identify individual quantities.

Principles are conceived of as equations (and are referred to in the paper as "equations"), whose elements are slots to be filled by quantities from the problem. This matching of equation slots with problem quantities is done by matching their names. Thus a quantity named "t", cannot match a slot named "v". The names assigned to quantities fulfil the function of MECHO typing information and quantity names at the

same time. Clearly, it is necessary to get the names of quantities right if a problem is to be solved. But a difficulty arises in the eventuality that a problem involves multiple contexts. For example, consider this question:

> A tram accelerates from rest with an acceleration of 2 m/sec/sec. It travels for 150 metres, and then travels at constant velocity for 20 seconds. How far has it gone?

In this problem, it is necessary to regard the speed of the tram after it has travelled 150 metres as "final velocity", with regard to one interval, and "initial velocity" with regard to a second. This makes the simple rules for symbol assignment described in the paper unworkable. The more complex the problems become, the more such situations will arise, and the greater will be the difficulties caused by the attempt to use the quantity names as both types and tokens.

3. Grouping Principles by Methods

The way the authors constrained principle selection during forward inference was to group principles into "methods", such as kinematics, energy, or force equations. The data seem to confirm the utility of this process. However, this can be interpreted in two ways:

(a) Experts group their principles into methods- this is how they solve problems.

(b) Problems are usually constructed so that they only need principles from the same method for their solution.

If humans are asked to solve problems requiring only one method to solve them, the difference will not be apparent. But it is in principle possible to distinguish between the two interpretations by constructing problems requiring principles from more than one method. The belief that experts rely on principle grouping to constrain search would imply that they made false starts in such problems, and needed to reject equations after they had generated them. The idea that "methods" are an artefact of problem construction suggests that this would not happen.

Until experimental data is collected to test this out, the use of "methods" has a dubious epistemological status.

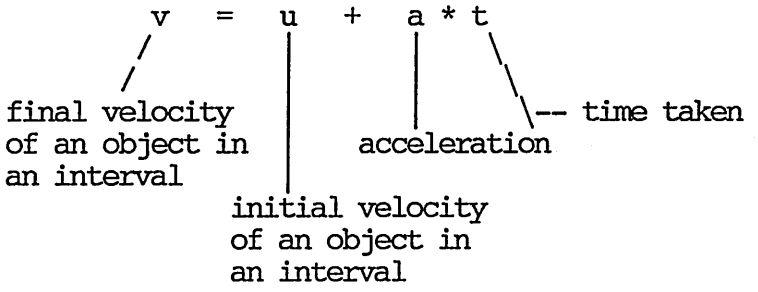
4. Where do Contexts come from?

Contexts are important to both the novice and expert models, but no mention is made of how they are identified. They might be assumed to be identified in the problem, or they might be constructed by the model- it is not stated. For the kinematic and dynamic problems described, a "context" is simply a time interval at whose ends an object has known quantity values- there is no need to search for the best of a number of possible contexts.

In more complex problems, however, considerable search and construction might be necessary to define an appropriate context. The "Testbed" problem described earlier, for example, requires use of several contexts, each of which is an < object, direction > pair. Since there are not only several possible objects for which to resolve forces, but an indefinitely large number of directions in which forces could be resolved, the selection of an appropriate context is a non-trivial task. If the "expert model" has no mechanism for doing this, it is likely to run into difficulties on problems more difficult than those it has been demonstrated on.

5. Use of Variadic Formulae

Principles in the domain of kinematics look like equations with a fixed number of well-defined slots. Thus:



This formula has just four slots. From the way the paper is written, the authors appear to feel that all principles are of a similar form- they even refer to principles as "equations". However, many physical principles are variadic- that is, they have a variable number of slots. For example, the "Conservation of Energy" principle has as many slots as there are types of energy affecting the system- something that will vary from question to question.

A process of matching quantity names to slot names is straightforward when formulae have fixed slots, but is

problematic in the variadic case. No mechanism for dealing with this is mentioned.

6. Combinatorial Explosion of Context Relations

The authors mention one specific piece of meta-level information that has been incorporated into the model. This is the knowledge that if two bodies are at the same places at times t1 and t2, they have travelled the same distance in the interval (t1,t2).

This corresponds to the sort of result produced by the "inference mechanism" in MECHO. The MECHO team found that the use of equality relations in meta-level inference led to explosive search in any but the simplest problems. In general, reflexive and transitive predicates like "in the same place as", and "is distant from", led to severe difficulties with explosive search. This was dealt with by a sophisticated inference mechanism (Bundy, Byrd and Mellish 1982), which relied on meta-level information about the relevant predicate- such as whether it was functional, transitive, reflexive, and so on.

The "expert model" does not apparently do this- and it is not clear whether such processing could be incorporated into the model, or even written in a production system. From the psychological point of view, it is implausible to propose a mechanism that may require extensive search as a model for human behaviour which does not.

Related Models: PH-100

A slightly different approach is provided by the PH100 model (Larkin 1979). This is a production system model that solves problems in dynamics- including problems involving more than one context. The same model is used to attempt to model both expert and novice protocols in the domain.

As in the previous study, the model uses a restricted working memory coupled with an unlimited database termed a "paper memory". As before, the expert version used forward inference and the novice version used backward inference. Test data based on the order of principle generation supported this claim.

Structure of the Model (Expert Behaviour)

- the model starts with a coded problem representation referred to as a "sketch". Larkin says: "This sketch is intended to represent, in computer-readable form, information which human solvers commonly abstract from the English problem statements and immediately write on their papers".

- the model then constructs a "physical representation", for the problem. This involves labelling every body mentioned in the problem, and listing for each body the directions of all forces acting on it.

- next, the model generates a "basic equation", for each body. This corresponds to an application of the "resolution of forces" principle in an undefined direction, yielding an equation with a defined number of terms, some of which will be unknown quantities.

- some unknown quantities are removed from the basic equations by the use of such physical principles as the friction law or Hooke's Law.

- when all possible substitutions have been made, the model then solves the remaining simultaneous equations.

Structure of the Model (Novice behaviour)

- the model starts with a "sketch".

- It now looks for a principle that involves the sought unknown, such as "Resolution of Forces"

- The model attempts to instantiate the chosen principle with quantities from the problem. This should involve using the quantities relating to the relevant body, and hence should be equivalent to the "Constructing Physical Representation" phase of the expert behaviour version. The novice behaviour version is unable to do this, however.

The fall-back procedure for the model when it cannot list the forces on a body is described by Larkin thus: "students generate forces corresponding to 'active' objects". Examples of 'active' forces included gravity, and the force due to a stretched spring. Whether reaction and tension in an inextensible string count as 'active' is not stated. In the example shown, 'active' forces acting in different directions

on different bodies were added together to give the F term of the Resolution of Forces principle "F = M * a".

This suggests that the novice behaviour model has the rule: "Add any active forces in the problem together to obtain F".

- The model now tries to substitute for unknown quantities in the equation. Again, it does this in a rather random fashion. When needing to find a mass to instantiate the F = M * a principle, it apparently chooses masses from the problem at random. Thus it makes errors, some of which novices also make on occasion.

Features of the Model

The features of this model which distinguish it from the previous ones are:

(1) The model starts with a "sketch", rather than simply a formalization of the original problem. This is claimed to be a significant aid to the problem-solver.

(2) The model constructs a "physical representation" for every body before selecting any principles to use.

This corresponds to a free-body diagram for each body mentioned in the problem.

(3) Every physical representation is used to generate a Basic Equation. This corresponds to using the F = M * a principle. Apparently, no other principle (conservation of energy, kinematics, etc.) will ever be used. We are not told how the direction of resolution is selected. Nor is there any indication of the order in which Basic Equations are generated.

(4) The novice is represented by a model that uses backward inference, and has trouble in instantiating the base equation.

Its method of instantiating is not specified, but I assume that it is done by type matching quantities in the sketch at random. Presumably the type matching is done by name, since separate typing information is not mentioned.

Significance of the Features of the Model

(1) The Sketch

This consists of a list of qualitative statements about the problem. Some of these correspond directly to statements in the problem, whilst others, such as information about directions, do not. This is not a sketch in the normal sense of the word, as it contains no metric information. Whether this is important or not will be discussed later.

(2) The Physical Representation

The model constructs a "physical representation", or free-body diagram, for every body mentioned before choosing principles or generating equations. This makes it impossible for this model to guess principles before the whole problem description is read in. Since the model not only generates physical representations, but writes them in its "paper memory", this facet of the model should be testable in practice. If an expert is given a problem for which one body does not require a free-body diagram (or its equivalent in terms of diagram

labelling), this model predicts that he would nevertheless create one.

For example, consider the following "garden path" problem (forces are measured in Newtons, or 'N'):

> A seesaw ABCD is pivoted at B, its centre of mass. A child weighing 200 N sits at C. The seesaw is held horizontal by a light inextensible string attached to D, which is vertical at D and passes over a smooth pulley, P. The other end of the string, S, is attached to a rough block of mass 80 Kg. which rests on a rough sloping plane XY. The coefficient of friction between the block and the plane is 0.5. PS slopes at 50 degrees to the horizontal and XY slopes at 30 degrees to the horizontal. Find the tension in the string.

This model predicts a free-body diagram would be generated for both the seesaw and the block on the plane. As Larkin has not tested such predictions, the justification of this feature must rest on its plausibility alone.

(3) Principle Selection

It is simply not true that every dynamics problem is solved by the principle F = M * a.

The problem described by Chi, Feltovich and Glaser, and solved by "Expert J.L." is an example of a dynamics problem solved by Conservation of Energy. The model predicts that Base Equations will be generated in a random order- and yet the evidence of the previous paper is that they are not.

(4) Novice Errors

If novices instantiate basic equations at random with quantities that appear to have an appropriate type, the model can serve as a generator of expected errors. Larkin claims the model has generated a number of errors that novices have displayed. However, the model generates a number of possible errors for each problem it attempts to solve. Do all of these occur in novice solutions? There is no data on this question. A model which predicts errors that never occur is not necessarily better than one which does not predict any errors at all.

Related Models - Expert and Novice Performance in Solving Physics Problems

Larkin et al (Larkin, McDermott, Simon and Simon 1980), produced a paper entitled "Expert and Novice Performance in solving Physics Problems", which was intended to summarise the state of knowledge about expert performance in Physics.

This paper did not present a new computational model, but referred to previously described models (especially by Larkin) to illustrate the theoretical points made.

The authors looked at Physics problem-solving as a special case of generalised problem-solving, and claimed that a common foundation underlay all problem-solving skills. As the authors put it:

"There is reason to believe that expertness has much the same foundations wherever encountered. As in genetics, we learn much about all organisms by studying a few intensively. Chess, algebra and physics are serving as the Drosophila, Neurospora and E. Coli of research on human cognitive skills."

They do not say what the reason is for believing that expertise in chess and physics have the same foundations.

The authors of this paper have acted with commendable boldness in defining exactly what psychological claims they are putting forward. They have not been seduced by the "gee-whiz" approach of writing a clever program and then describing how it works without attempting to establish its connection to cognitive psychology. Neither have they mudged about with a haze of vague statements equally compatible with any one of a variety of different implementations. By putting their conclusions with such estimable clarity, they have laid the grounds for future progress; whether their conclusions prove to be accurate or not.

Psychological Claims

(1) Short-term memory has a capacity of from four to six items, where an item is any stimulus recognizable as a single unit. This includes referencing complex problem-solving procedures from permanent memory, which can be indexed in short-term memory as a single item.

(2) The indexed memory is organized as a large set of production rules. Their condition parts are compared with the contents of short-term memory, and when a match is obtained, their action parts are executed.

(3) The data structures in the human memory which the productions manipulate consist of list structures.

(4) Experts solving mechanics problems construct representations equivalent to sketches of the objects in the problem. These representations are node-link structures isomorphic to physical diagrams.

(5) Experts solve problems about four times as fast as novices.

(6) Novices solve problems by backward inference.

(7) Experts solve problems by forward inference.

(8) The disadvantage of backward inference is that the management of goals and subgoals may occupy considerable time, and overload short-term memory.

(9) Experts merge the stages of stating a principle, instantiating it to an equation, substituting values into it and solving it. Novices do not.

(10) The novice works directly from the objects given in the problem, whilst the expert works from an intermediate representation of "physics quantities".

(11) An expert's attention (i.e. current goal) is controlled by the knowledge in short-term memory only, and not by the identity of the last production executed.

Review of Psychological Claims

(1) The concept of a limited short-term memory for around 4 - 6 items developed originally from research on the memorization of meaningless symbols. The authors have made the plausible inference that this memory limit applies to pointers to data structures residing in long-term memory as well.

Although such a claim is in principle testable, it is hard to see how it could be tested apart from a particular implementation, since different production system models might make different demands on short-term memory. The authors are unsure as to whether the current task goals have to be held in the short-term memory or whether they are held elsewhere.

(2) and (3)

These are very strong claims for the psychological accuracy of production systems, going way beyond the belief that they are "fairly transparent representations of verbal theories". As such, it is inappropriate in the present state of knowledge to ask for proof or refutation of such claims. If it is felt that the concepts are fruitful, they will become widespread. If no useful results appear to flow from them, then they will fade away.

This approach provides a challenge, however, to the builder of cognitive models. If the highest level of their program organisation is claimed to be a psychologically accurate model, and the base level of the language structure is held to be psychologically accurate as well; then presumably all the intermediate levels of structure must reflect psychologically valid processes too.

It would then be a reasonable question to ask the author of a computational model what was the psychological basis for any single production in the model. This is certainly not a question I would ever like to have to answer.

(4) Some examples of "sketches", as "node-link structures", are given; and others can be found in the previous paper by Larkin. The motivation for introducing a sketch is stated thus:

> "It is easy to see why pictorial representation is convenient. Much of the difficulty with mechanics problems lies in understanding the spatial relations among the objects. Moreover, the pictures that are drawn can be highly stylized, abstracting away irrelevant information in the English problem statement."

However, there is a significant difference between the node-link structures the authors call "sketches", and the drawings made by human problem-solvers: human sketches have metric properties, and node-link structures do not.

The essence of creating a sketch for a person is that a qualitative relationship is processed to produce a particular metric instantiation of it. Whether the problem-solver knows the measurements of his sketch or not, the lines on the paper have particular lengths, and intersect at definite angles. This has some advantages not mentioned by the authors.

For instance, impossible descriptions cannot by sketched, and are therefore seen to be impossible. Consider this description of this two-dimensional framework:

distance (A, B, 3)

distance (B, C, 3)

distance (C, A, 3)

distance (A, P, 4)

distance (B, P, 4)

distance (C, P, 4)

Any procedure that tried to draw a metric sketch of that would fail. Of course, human problem-solvers are not able to distinguish small differences in length by using freehand sketches, but gross differences between label values and their representative components in the sketch will be readily apparent. Any program that could maintain a representation such as that given above; cannot therefore be said to "sketch" the system it deals with.

The authors mention a similar problem in the paper, previously investigated by Paige and Simon (1966):

> A board was sawed into two pieces. One piece was two thirds as long as the whole board. It was exceeded in length by the second piece by 4 feet. How long was the board before it was cut?

The contradictory nature of this problem statement would be exposed if a sketch was constructed with instantiations of metric measurements. If we assume that the phrase "It was exceeded in length by the second piece by 4 feet" cues a test to see if the second piece is longer than the first, then we have a mechanism for identifying the contradiction. the authors state that when asked to solve this problem, some students generate and solve the equation:

```
2 * x / 3 + 2 * x / 3 - 4 = x
```

- as if they had read that the second piece was four feet shorter than the first. They explain this by assuming that these students have already solved the "correct" equation:

```
2 * x / 3 + 2 * x / 3 + 4 =x
```

obtaining the solution x = -12; and then backtracked to try a different equation, on the grounds that no board is -12 feet long.

This explanation assumes an ability to solve such equations mentally, in which case why should they write down any equation at all? For a survey of psychological investigations into the use of mental sketches or models, see Johnson-Laird (1983).

(5) The fact that experts solve problems four or more times as fast as novices is interesting. It would be even more interesting if we knew what stage of the problem solution they did faster. Chi et al. found that experts took longer than novices to categorise problems- so some parts of the solution must be done more than four times as fast. An analysis of which stages took longest would be a useful indication of what stages involved much processing, and what stages involved little.

The authors are convinced that deciding what step to take next is what takes novices a long time to solve problems. By implication, one would assume that deciding what step to take next is something experts are fast at. This is compatible with

the proposal that experts can tell before generating an equation what quantity it will solve for and what quantities it will introduce. This enables experts to select an appropriate principle swiftly, while novices, faced with many possible principles (and often, many possible contexts also), find it difficult to do this without actually generating the equation.

(6) It seems now well-established that novices solve problems by backward inference.

(7) The claim that experts solve problems by forward inference rests only on the order of principle used. Therefore it is too strong, and should be replaced by the claim:

"experts use forward inference or planstacking".

Regarding the "forward inference" claim, the authors say:

> "This was a bit surprising, since it is usually thought that working backward is a more purposeful and sophisticated strategy than working forward. The answer to the puzzle may be that experts work forward on easy problems, where their experience with the problem domain assures them that,

without any particular planning, solving all equations that they have enough information to solve will lead them quickly to a complete understanding of the situation, including finding the particular quantity they are asked to solve for."

(8) The authors claim that the disadvantage of backward inference is that the management of goals and subgoals may occupy considerable time, and overload short-term memory. This is a very subtle and interesting point. In fact it is a claim for the psychological accuracy of production systems, because in a production system, goal management is organised in such a way that this claim about goal management is true.

Explicit data structures relating to goal scheduling are stored in memory, and can be read by productions. But in other computational formalisms this is not so. In PROLOG, goal scheduling is handled by the interpreter, and no data structure relating to goals is asserted into the database or passed as a parameter. A backward inference mechanism in PROLOG requires no more search or memory than a forward inference mechanism.

Therefore this is a claim about the internal representation language of the human problem-solver: but what claim is being made? Is it the claim that human processors keep track of goals via independent data structures (which would violate the applicative constraint proposed by Van Lehn)? Or is it just the claim that backward inference is harder than forward inference for unspecified reasons? Either way, it is hard to see how such a claim could be tested.

(9) Certainly at the level of behaviour, experts merge the stages of stating principles, instantiating them, substituting values and solving equations. The authors appeal to the distinction between compiled and interpreted code for an explanation.

An alternative explanation might be that for an expert, the selection of a plan is preceded by a stage in which meta-level inference is used to analyse which quantities the equation to be generated will contain. Also, the equation to be generated will contain at most one new unknown quantity, since it will be generated in the order of a forward inference process.

Thus when the principle is instantiated to form an equation, the expert knows in advance which quantities will be involved,

and will already have available a substitution for one of them derived from the previous equation.

This extra information allows the various stages of equation generation and solution to be merged into one. The novice, on the other hand, has not the sophisticated meta-level inference mechanism that will tell him in advance what quantities will appear in the equation. His only method of finding out is to generate the equation and then examine it. Nor can he do instant substitutions if he is using backward inference; nor would it be sensible of him to incorporate equation rearrangement and solution steps until he had checked whether the equation contained the right quantities.

Of course, such an explanation is not incompatible with certain parts of the process being compiled whilst others are interpreted.

(10) It is claimed that the novice works directly from the objects in the problem, whilst the expert works from an intermediate representation of physics quantities. This claim can be interpreted in two ways:

Firstly:

That the expert constructs an intermediate "physics" representation of the whole problem before applying principles to generate equations. The novice does not identify the physics quantities until they are required. Thus the novice identifies the physics quantities relevant to the sought unknown first of all, and does not identify the others till later, if at all.

The main difficulty with this idea is that of reconciling it with the findings of Chi, Feltovich and Glaser, that experts guess or select principles before the whole problem has been read in- and therefore before the intermediate representation has been completed

Secondly:

That experts use principles to generate equations on the basis of the physics quantities relevant in a given context, while novices generate equations on the basis of the objects.

So the experts would be using the forces, accelerations and so on, while the novices would be dealing only with blocks, strings, and physical objects like those.

It is quite impossible that the novice could generate equations without using physics quantities- because this is what equations consist of. So the claim must be that the principle used is selected on the basis of objects rather than quantities.

No-one has actually produced a model of novice behaviour that is able to select principles to solve problems by consideration of objects and not quantities. Nor has anyone explained how it could be done. The sought unknown is a quantity, not an object; and it is possible to ask questions involving the same objects, but different sought unknowns, such that the questions require different solution principles.

Such questions cannot be answered at all by problem-solvers who do not relate principles to quantities- presenting a severe difficulty to anyone who claims novices do not choose principles by analysing physics quantities.

There is a clear reason for thinking that novices concentrate more on objects than experts- the problem categorisation data of Chi et al. This can be adequately explained by supposing that experts possess a mental mechanism (i.e. - one that does not rely on external or paper memory) capable of identifying the correct solution principle. Novices, not possessing such a mechanism, are unable to classify problems in this way, and so classify problems in terms of objects by default.

Such an explanation is not incompatible with an ability to build up representations of physics quantities, and reason with them in order to select a principle to use.

(11) The final claim is that an expert's attention, or current goal, is controlled only by the contents of short-term memory- there is no explicit ordering of productions. This is an extension of the earlier claims for production systems, and the same remarks apply to this as to them.

The ABLE System

In "Skill Acquisition for Solving Physics Problems" Larkin (1979) describes a computational model called "ABLE". This is a production system which initially solves physics problems by backward inference, but which learns from its experience other methods of solution which Larkin compares with those of experts. The starting version is termed "barely ABLE", the partially experienced version "slightly ABLE", and the expert version "more ABLE".

Larkin does not in this paper make strong claims for the psychological validity of production systems, but justifies her use of such a system in these words:

"Production systems have the following major virtues:

(1) They involve plausible and parsimonious assumptions about the human information processing system. All permanent knowledge is entered homogeneously without a priori distinctions between (for example) declarative and procedural knowledge.

(2) Their architecture makes modelling learning easy. All knowledge is encoded in relatively independent productions, and adding knowledge just means adding productions. Thus knowledge can grow through small increments.

(3) Productions can be written so that each corresponds to a bit of knowledge that makes psychological sense. Thus they are fairly transparent representations of verbal theories."

The Model

ABLE begins with a "sketch" of the problem (in the sense used by Larkin, McDermott, Simon and Simon). Then it labels all the physics quantities in the representation with names that will match the names in Physics principles. So initial velocities will be marked "Vo", accelerations "a", and forces due to gravity "Fg". Each quantity mentioned in the sketch is labelled "known" or "sought".

Barely ABLE now looks for a principle (which has the form of an equation involving unbound quantities), relating a sought quantity to other quantities. It binds the slots in the equation to quantities in the representation, and lists new sought quantities if needed. It assumes an equation with one variable is soluble, but it doesn't actually perform the algebraic work of solving it.

Because barely ABLE selects principles only on the basis of the quantities they contain, it often selects a useless principle and needs to backtrack. Backtracking is constrained by the limited size of the short-term memory; and solutions requiring substantial backtracking result in some of the contents of working memory being lost. This leads to circular behaviour or to an impasse situation in some problems. When ABLE generates a soluble equation (i.e. - one with a single unknown variable), it records the circumstances and the result. The "circumstances" are the known quantities, the objects involved, the desired quantity, and the principle used to find the desired value.

An automatic production is created and asserted into the production memory, to the effect that next time the same circumstances are present, the same result can be returned.

When ABLE solves a problem, it may create a new automatic production. So by experience, it builds up many automatic productions. When ABLE next solves a problem, the use of automatic productions takes precedence over the backward inference mechanism. This results in a slighty ABLE model with a few automatic productions, which starts by generating one or more equations in forward inference mode, and then returns to the backward inference method when no automatic production can take it further.

A larger number of automatic productions yields a more ABLE model, working all the time by forward inference. In this case, the backward inference mechanism hangs around in the background like a wallflower at a party, waiting in vain to be asked for its opinion.

All the physics principles except one are encapsulated in single productions. The exception is the so-called "Superposition Principle", which states that the resultant force on a body is the vector sum of all the separate forces. Because this is not expressed as an equation with a fixed number of slots (i.e. - it is a variadic principle), it is hard for ABLE to acquire automatic productions involving this principle.

The difficulty is that if an automatic production is created for a situation involving a certain number of forces, the production may fire on another question where the situation actually involves more forces than the automatic production allows for. So an equation will be generated that omits terms corresponding to particular forces.

The Data

The model was compared with scripts from experts, novices, and intermediate level problem-solvers. In terms of the order of equations generated, barely ABLE and more ABLE provided good descriptions of novices and experts respectively.

The results from the intermediate level problem-solvers were interesting. They frequently showed one or two equations being generated in a forward order, followed by the completion of the problem by backward inference. This is a striking piece of evidence in favour of Larkin's model, because the ABLE system predicts exactly such behaviour. For an intermediate level problem-solver, ABLE assumes that they will possess some schemata, and so will be able to start solving problems by forward inference. They will not always have enough schemata to complete the problem solution, and so they must finish the problem by backward inference.

Larkin attempted to duplicate each of these intermediate level solution traces by tuning up ABLE with particular automatic productions to duplicate each trace. This was usually possible.

Some novice scripts included incorrect equations; some of which omitted forces, and others of which were incorrect in other ways. ABLE could generally duplicate the force omission errors by the use of appropriate automatic productions. Larkin wrote special purpose productions which were able to duplicate most of the other incorrect equations, but the paper does not state how general these productions were, or whether their existence would have led to different behaviour when solving other questions.

Larkin summed up the results of her study by saying:

> "Clearly such a primitive model fails to capture much of the richness of human knowing and learning. In particular, the primitive and algebraic problem representations are very limiting. However, the mechanisms of this very simple model account very well for the order in which principles are selected and used by human solvers of varying degrees of experience. In addition, the single principle that requires extra productions in the model and is prone to errors when the model learns, is the same principle that is almost universally misused by inexperienced human solvers."

(the principle that is almost universally misused being the Superposition Principle described on page 189)

Issues Arising

(1) The size of the space of automatic productions

(2) The adequacy of automatic productions in explaining expert behaviour

(3) The relevance of automatic productions

(4) Possible interpretations of intermediate level forms of problem solution

(5) The principle of superposition and alternative interpretations of the data

(6) Automatic productions and errors of superposition

The Six Issues Considered:

(1) If we ascribe an automatic production to every possible sought variable, in conjunction with every possible selection of known variables, for every possible system of objects, we end up with a very large number of automatic productions

indeed. This is not in itself a problem, as there is reason to believe the human mind is capable of memorizing a very large number of meaningful patterns (deGroot 1978).

However, it is likely to be significantly larger than the number of questions and examples in a textbook (else it would be impossible to write a textbook without repeating questions). If the number of such possible productions is larger than the number of practice questions needed to teach a novice to be an expert, then either there will be some questions the "expert" cannot solve in an expert way, or there are situations the expert solves in an expert way that he had no opportunity of acquiring a production for.

The ABLE mechanism could in principle be tested by estimating the number of possible automatic productions, and also the number of problems a novice will be exposed to before he becomes an expert. It the first number were greater than the second, a serious inconsistency would have been shown. If the two numbers were approximately equal, this would be suggestive evidence in favour of the model.

(2) A related point to the first is the necessary incompleteness of the automatic productions. It is possible to

devise (rather funny-looking) problems that involve contexts no expert is likely to have met. For instance:

Find X in this situation. All side weights hang from pulleys.

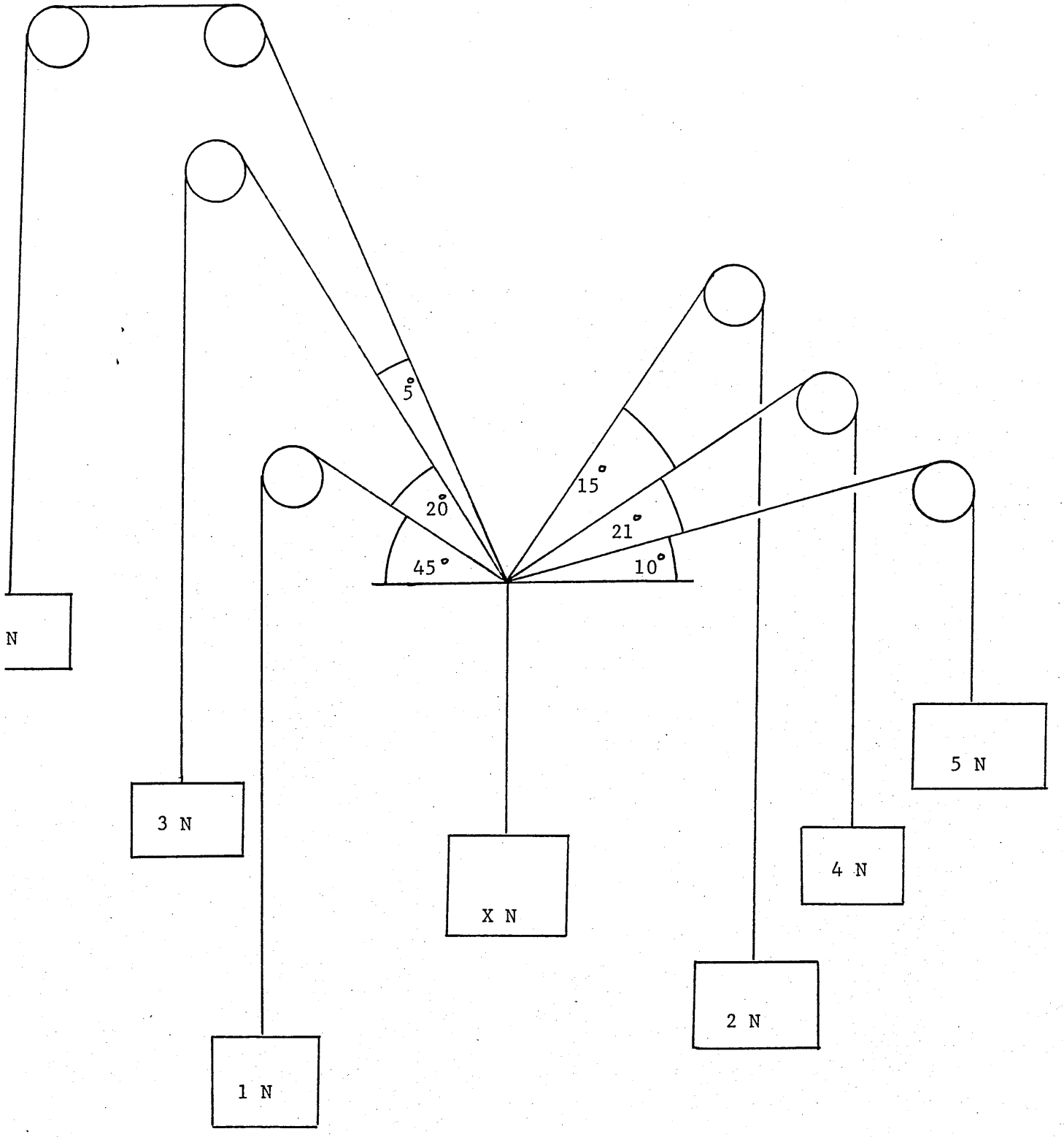


Figure 5. The Pulley Problem

If the ABLE mechanism is the method used by experts, one would not expect them to have a forward inference method of dealing with such problems. An expert who does deal with problems involving contexts new to him by the same method as other problems would therefore not be using an ABLE-type mechanism.

(3) Larkin admits that sometimes the expert version of ABLE cues irrelevant productions.

> "If there are many automatic productions from several different domains of physics, then productions from different domains tend to act incoherently without progressing efficiently towards a single solution".

This expresses in a nutshell the difficulty with forward inference models in general- how do they know which principle to select? This difficulty could be overcome by dividing up the automatic productions into groups corresponding to the "methods" of Larkin et al.- but only if it could be guaranteed that a production from the correct method was selected first of all

A problem involving more than one context, and requiring a principle from a different method for each context, could not be efficiently solved by such a model. The limitations of short-term memory referred to by Larkin to account for novice difficulties might even prevent an expert model from solving such problems at all.

(4) The strongest point of the ABLE model is its ability to duplicate the principle ordering of intermediate level problem-solvers.

Could this be duplicated by another method? At first sight, one might expect a planstacking problem-solver that has generated one soluble equation to complete the solution in the same way- because to generate one soluble equation, it needs to build a complete planstack. If a problem-solver has built a complete planstack, there seems on the face of it no reason why he should not pop the stack and generate equations in forward inference order.

A possible explanation of this intermediate level behaviour might be that due to memory limitations, the novice problem-solver is only able to use a planstack of limited size. As more plans are pushed on the stack beyond a certain limit,

those at the bottom of the stack are lost. Thus when the conditions for popping the stack are satisfied, only the top few levels of the stack are available. This accounts for a few equations generated by forward inference, followed by an impasse. The obvious thing for the problem-solver to do next is to regrow the planstack from the beginning again- but this hypothesis is incompatible with the data presented by Larkin.

The equation ordering presented as typical of intermediate level problem-solvers can, however, be explained in the following way:

Intermediate level problem-solvers have a limited size of planstack. When the planstack is full, the addition of new plans causes the loss of the oldest ones. When the stack is popped, a few equations are generated by forward inference and then an impasse is reached, because the oldest plans no longer exist.

When this impasse is reached, the problem-solver abandons planstacking as a method of problem solution, and defaults to backward inference. This results in the rest of the relevant equations being produced by backward inference, In short, planstacks leak at the bottom when full, and a leaky planstack is not trusted. This would appear subjectively to the novice

as if he had tried to plan a solution, but found that the plan failed to work. At this point, he would revert to what seemed to him the more primitive but reliable method of backward inference.

This would explain the data described above. Without experimental testing, this hypothesis rests only on its plausibility as an evidential foundation, and as yet no such study has been carried out.

One conclusion can be derived from this adaptation of the planstacking model though- it implies that either (1) experts have larger planstacks than novices (which does not seem compatible with the invariance of short-term memory assumption), or (2) a "plan", as handled by an expert problem-solver consists of fewer "memory units" than a "plan" as handled by a novice. Of course, the elements of a plan are necessarily placed on the stack in conjunction- a context without a principle having no procedural meaning to the problem-solver. But that does not rule out the possibility of some form of plan compaction process which enables an expert to use fewer "memory units" to store a plan than a novice. This looks a more promising line of approach- one would expect the conceptual units used by an expert to be packaged into higher-level chunks than those of a novice. Assuming such "packaging"

is dependent on experience, it can be argued that the frequent association together of elements from a plan leads to a structure being created in long-term memory which combines them together in a single unit which can be manipulated by short-term memory.

What sort of elements could be frequently and reliably associated together in this way?

Let us consider the testbed problem mentioned earlier. One of the plans used was "resolve forces vertically at b". This contains the elements:

"Resolve forces"
"In the vertical direction"
"At b"

If this is held on the planstack as separate items, it will take up three slots in short-term memory. However, if "Resolve forces in the vertical direction" is held as a single unit, two memory slots will be needed instead. This could not be compacted to a single unit because the "b" is unique to this particular question, while "Resolve forces vertically" is an operator frequently used in similar problems.

On the other hand, it is hard to see how plans like "use the formula:

$$v^2 = u^2 + 2 * a * s \quad \text{for the interval (t3, t4)",}$$

could be compacted in this way, because the interval is particular to the question, and the rest of the plan is single data item anyway.

If this explanation of the experts' greater planstack capacity is accurate, it would suggest that an expert has a greater apparent size of planstack when solving some domains than others. In principle this could be tested- but the interpretations of the tests would depend on the assumption that an impasse in planstacking causes default to backward inference, as well as the assumption that planstacks leak at the bottom.

So to extract any sensible conclusions from data on how many successive forward inference equations an expert can generate before resorting to backward inference in a particular domain is hard. It involves piling the Ossa of plausible assumption upon the Pelion of vigorous extrapolation of results. And of course it is a matter of personal taste whether the assumptions given here are found to be plausible or not. This is the

weakest point of the planstacking hypothesis as regards its observational adequacy.

(5) Some novices omit terms involving forces from equations. Larkin explains this as difficulties with the principle of super-position of forces.

An alternative explanation is that every time a force is omitted, it is because the novice has particular weakness involving forces of that particular type. So the omission of a friction term could be explained as "not knowing about friction", rather than "not knowing about superposition".

From the behavioural point of view, the Larkin hypothesis would suggest that in solving a number of problems involving the superposition principle, novices would omit different forces on different problems. The alternative view would suggest that if a force is omitted in one problem, then forces of the same type, and not others, will be omitted when solving subsequent problems. Some experimental evidence supporting the "omissions related to type of force" approach will be dealt with later.

A problem with the "superposition is difficult" theory which applies to the ABLE model is that ABLE starts with what Larkin calls a "sketch", and then proceeds to label all the quantities that affect objects in the sketch. The purpose of the sketch is supposed to be to clarify the spatial relationship of objects and quantities- so how is it that forces can get missed out?

(6) Superficially, it might seem that the biggest weakness of ABLE is that starting from a novice model that solves problems correctly, it generates automatic productions that are supposed to correspond to an intermediate level problem-solver- yet these very automatic productions can lead to errors.

So the process of developing from novice to expert seems to consist of starting without making mistakes, and then learning to make them. Neither Larkin nor any other worker in this area has suggested that this is in fact so. I do not believe this is a serious difficulty. Enough work at the implementation level should be able to ensure that automatic productions apply only in contexts where the relevant quantities are the same as those in the problem which the production was derived. Such a technical improvement would not affect the basic structure of ABLE, nor the validity of the psychological claims based upon it.

An Alternative Model of Human Problem-Solving: The Planstack Model

The basis of this model (Priest 1986), is an attempt to conform with the experimental data presented in the papers discussed earlier in this chapter. No claims are made for the psychological validity of any representation language, other than the claim that certain data structures can be treated as "single units" when considering their demands on short-term memory.

A strong assumption underlying in the model is that people find some things easy- such as dealing with qualitative representations of data structures; while other things are difficult- such as manipulating explicit equations. It is also assumed that people can master procedures consisting of a large number of sequential stages more easily than procedures involving significant amounts of search and backtracking. Therefore it will be taken as psychologically plausible to propose processes involving many steps of qualitative inference with familiar data structures. It will be taken as psychologically implausible to propose processes involving a lot of search and backtracking, or internal manipulation of equations.

As regards the differences between experts and novices, it will be assumed that the skills possessed by experts but not novices are exactly and only those processes which are not taught explicitly by textbooks or on courses. These will be identified as the ability to tell what quantity will be found by implementing a given plan; the ability to store and manipulate plans internally; and the hidden curriculum assumption of non-redundancy.

The arguments used in the previous sections can now be used to define an alternative model for physics problem-solving. This model is based on a planstack. Plans are stacked as they are generated by meta-level inference, until the planstack is full. Then the planstack is popped, and as every plan is popped from the stack, it is used to generate an equation. This process accounts for the observations that experts generate equations in a forward inference order. If we assume that experts find planstacking so easy that they can do it subconsciously, this also explains why experts are able to classify problems by solution method before solving them- the classification is a description of their planstack. It also explains how experts can select the right principle with which to generate their first equation.

A plan is selected by meta-level inference about the quantities related by different principles, and the quantities appearing in different contexts in the problem. Such reasoning can also be performed by MECHO and ABLE, so there are definitely adequate methods of formalising this reasoning computationally.

Experts can analyse the consequences of their plan without generating equations; and form a complete stack before producing any equations. Novices, however, cannot tell what quantities an equation would contain until they have generated it- so they pop the stack after every plan has been pushed onto it. This leads to a forward inference order of equation generation for experts, and a backward inference order for novices.

That is the core of the model. Other details are required to justify other experimental findings. To explain the ability of experts to guess the solution principle before the question is completely read in, the model calls for a shortlisting of principles to be developed during the reading process. This relies on the Hidden Curriculum Assumption that all quantities in the question are relevant.

To explain the action of problem-solvers in drawing sketches, and in the actions of some people in reparsing contradictory

questions in order to produce a meaningful representation, the model calls for a sketch to be produced. This is an arbitrary metric instantiation of the problem description. The model does not address the question of whether the sketch is expressed externally or not- it is compatible with either implementation. Since the metric instantiation of the sketch is arbitrary, it is possible that a chosen sketch will not be consistent with the problem description- in which case it must be possible for the problem-solver to backtrack and produce an alternative sketch if needed. Likewise, the question involving the sawn board described earlier shows that it must be possible for the problem-solver to back up its execution state in order to reparse the question. The planstack model described here does not deal with parsing natural language, but it requires that the parser be capable of alternate parses of a question when back-tracking. We shall assume that experts incorporate "consistency tests" in their behaviour via which they may identify contradictions in the sketch, and initiate backtracking. The novice does not possess such tests.

Experts use plans to generate equations differently from novices. Before expressing an equation externally, they substitute for quantities isolated in the previous equation. They also isolate the quantity the plan was formed to find. Novices do not do this. It is obvious why experts substitute

and novices don't- since experts generate equations in forward inference order, the previous equation will contain just the quantity required to substitute into the current one. If that quantity had been isolated when the previous equation was expressed, it will be easy to substitute for it in the current equation. This is why substitution is possible and isolation desirable.

There is, however, a problem with the isolation subprocedure. We have assumed up to now that the problem-solver finds qualitative inference easy and equation manipulation difficult. This is why experts build up a planstack and generate equations in forward inference order instead of using the simpler backward inference control strategy. This is also why experts write equations down, while their qualitative meta-level inferences and planstack are strictly internal. Even when an expert writes down the name of the principle he uses, this as a comment to guide the interpretation of the equation, not because it is needed on its own. So how does the expert manage to perform internally an equation manipulation task like isolation of a quantity? There are three possible ways it could be done, and the planstacking model of search control does not itself discriminate between them.

(1) The principle is stored in several different equation forms. Thus, where the novice might have the kinematic principle:

$$v = u + a * t,$$

the expert might maintain the representations

$$v = u + a * t$$

$$u = a * t - v$$

$$a = \frac{v - u}{t}$$

$$t = \frac{v - u}{a}$$

as equivalent possible forms. Then when the principle was used to generate an equation, the form used would depend on the sought quantity.

(2) The principle might be stored in a qualitative form that was easier to manipulate than an equation. The isolation procedure might be applied at this stage, and the equation generated from the rearranged qualitative form. Thus, in the testbed problem, the plan "Resolve forces vertically at the string junction", might first be instantiated to:

Vertical Component of tension1
+ Vertical Component of tension2
+ Vertical Component of tension3 = 0

In this case, tension3 is the sought quantity. So the qualitative form could be re-represented as:

Vertical Component of tension3 =
- Vertical Component of Tension1
- Vertical Component of Tension2

Then each of these qualitative terms could be instantiated to form the equation.

(3) It might be possible for an expert to remember and manipulate equations over short time-scales. There does not seem to be any direct evidence of this in the work described here.

These different approaches will be considered later, in the context of errors.

Whether novices are capable of such operations is a moot point- there is a reason why they wouldn't use them even if they could. When a novice generates an equation, he may need to reject it and backtrack. So it is pointless to perform processing until the equation has been checked for the quantities it contains. This necessitates writing the equation explicitly. There is not even much point in isolating the sought variable at this stage, because he will not be able to substitute it into the next equation. This is because of the backward inference order of equation generation.

Either of the alternative expert methods of equation generation is capable of explaining Larkin's finding that novices write down principles explicitly, and experts usually don't.

Experts don't maintain their principles in the same form as they express the equation, so an explicit representation would not be isomorphic to the equation. Thus it would be little help in checking the completeness of the generated equation. Novices generate equations isomorphic to their stored principles (the ones that look like formulae, that is). Consequently, they can use the visible formula as a check that no term has been omitted. This fits in with Larkin's finding that the most troublesome principle for novices to handle is Resolution of Forces; because no explicit formula can be written down, and so novices cannot perform syntactic checks for the omission of terms in their equations.

Intermediate Level Problem-Solvers

The final experimental finding to be covered by the planstack model is that intermediate level problem-solvers typically generate a few equations in forward inference order, and then complete the solution by using backward inference. This is explained as due to the overloading of the planstack, and the loss of its oldest contents. That accounts for the first equations being generated by forward inference, and then an impasse being reached. To account for the remainder of the

equations being generated by backward inference, we must assume that he does not build up a planstack twice for the same problem- if a planstack doen't work, he resorts to backward inference for the remainder of the question.

But why should a novice's planstack fill up when an expert's doesn't, even on the same problem? The easiest answer would be that experts have larger planstacks, but this does not sort well with data on other areas of cognitive psychology. Larkin herself refers to studies of expert chess players which concluded, contrary to expectations, that they did not have larger or better memories for chess positions than novices- they recognised a greater number of high-level patterns. Remembering a board position for an expert involved remembering the same number of items as a novice- but each item corresponded to a larger number of chess pieces.

Similarly, it is proposed that experts and novices have planstacks capable of holding the same number of primitive data items- where "primitive" is used to mean "counting as a single unit of memory". However, a plan to an expert will consist of fewer primitive data items than to a novice. Thus an expert can stack more plans than a novice. The compacting of plans into a smaller number of primitive data items must presumably depend on commonly used components of plans being stored in

long-term memory. Unfortunately, it is hard to infer the details of such a mechanism from the available experimental data, and no specific hypotheses about plan compaction are put forward (this may be regarded as another weak point of the model).

How many items can a planstack hold? I haven't the faintest idea. If one assumes that experts, as well as intermediate level problem solvers, lose confidence in a planstack that has leaked, one could get some kind of approximation by setting experts problems that required many stages for their solution, and seeing how many plans they could handle before reverting to backward inference. if one believed, however that experts generated replacement planstacks as required, then one would expect them to be able to handle problems involving unlimitedly large numbers of plans. The only sign of the planstack regeneration process would be a pause in the expert's working. After reading the question they would pause while they built up an initial planstack. Then they would pop the stack and produce a series of equations lickety-split.

At this point they would run out of plans, the oldest ones having leaked out of the bottom of the planstack. Then they would pause to regenerate the planstack. After this would come another burst of equations. The process would be repeated

indefinitely. Each burst of equations would result from the popping of a full stackload of plans. Each burst of equations would be followed by a pause shorter than the one before.

Experimental data might disconfirm the model, or provide information on how many plans could be held on an expert's planstack. A comparison with the number that could be held on an intermediate level problem-solver's planstack might find indications as to the compaction factor of expert as opposed to novice plan structure. Alternatively, such timing data might suggest that experts and intermediate level novices held the same number of plans in a stack. In this case, the difference between them would be that the intermediate level problem-solvers lost confidence in leaky stacks, and the experts didn't.

Specification of the Planstack Model

Control is guided at all times by the nature of the sought and given quantities in the problem. The Planstack model requires three types of control data to work with:

- Definite information on the sought / known status of quantities as specified in the question or derived from explicit equations

- Tentative information on what the sought / known status of various quantities would be if a plan were executed.

- Information as to which quantity a given plan was created to find.

As the planstack is filled up, various quantities are marked as likely to be known or sought after the execution of each plan. At any time there will be a list of such quantities, and for every plan there will be a quantity it was intended to solve for (but may not).

As the planstack is popped, the soughts and knowns derived from the generated equations are developed. These are also held in a list. Thus the control data requires four lists of quantitites, plus one quantity attached to each plan.

Qsoughts- a list of quantities definitely sought. This initially comes from the question. Every time an equation is generated, any new quantities introduced by the equation are added to the Qsoughts list.

Qknowns- a list of quantities definitely known. This initially comes from the question. Every time an equation is generated, the quantity from which it solves is added to the Qknowns list.

Psoughts- This is the list of quantities sought after a plan has been formed. The list is initialised to the quantities sought in the question, and every time a plan is formed which will introduce a new quantity, it is added to the list.

Pknowns- This is a list of quantities known after a plan has been formed. The list is initialised to the known quantities in the question, and every time a plan is formed, the quantity it will solve for is added to the list of Pknowns.

Pfound- Whenever a plan is formed, the quantity that will be found by the plan, referred to as "Pfound", is attached to the plan.

In order to avoid excessive complexity, the model is first described in the form it would take for an expert problem-solver. The complete version follows afterwards.

Planstack Model (Expert Version)

(1) If - Qsoughts in Qknowns
- stop

(2) If - another phrase left
- read it
- parse it
- sketch it
- add Qsoughts and Qknowns to Psoughts and Pknowns

(3) If - phrase just read
- collect the principles which involve every quantity mentioned
- put them at the head of the Principle List

(4) If - sketch just updated
- test sketch for consistency

(5) If - there's a Psought which is not in Pknown (the Pfound)
- search the principle list for a principle involving the Pfound

- search for a context where the principle can produce an equation involving the Pfound
- push the plan < principle, context, Pfound > on the stack

(6) If - plan just pushed on stack
- Add the Pfound to the Pknowns
- Add the other quantities involved in the plan to the Psoughts

(7) If - planstack not empty
- all Psoughts in Pknown
- pop the stack

(8) If - planstack just popped
- use plan to generate an equation
- use the previous equation to substitute for one of the Qknowns in the equation.
- isolate the Pfound

- write the equation
- and the Pfound to the Qknowns
- add any other quantities to the Qsoughts

Adding procedures to deal with the novice and intermediate level behaviour leads to the complete version of the model.

Planstack Model (Full Version)

In this version, procedures not possessed by novices are marked with a star.

(1) If - Qsoughts in Qknowns
- stop

(2) If - another phrase left
- read it
- parse it
- sketch it
- add Qsoughts & Qknowns to Psoughts & Pknowns

*(3) If - phrase just read
- collect the principles which involve every quantity mentioned
- put them at the head of the Principle list

*(4) If - sketch just updated
- test sketch for consistency

(5) If - there's a Psought which is not in Pknown (the P found)
- *search the Principle List for a principle involving the Pfound
- search for a context where the principle can produce an equation involving the Pfound
- push the plan < principle, context, Pfound > on the stack

(6) If - plan just pushed on stack
- Add the Pfound to the Pknowns
- *Add the other quantities the plan would use to the Psoughts
- mark the plan "tentative" if no quantities added to the Psoughts

(7) If - stack just pushed
- stack full

- forget oldest plan

(8) If - planstack not empty
- all Psoughts in Pknowns
or
- Last plan tentative
or
No confidence in planstack

- pop the stack

(9) If - planstack just popped
- plan is tentative
or
-no confidence in stack

- generate equation
- write equation
- check equation contains Pfound
- add the Pfound to the Qknowns
- Add any other quantity to Qsoughts

*(10)If - planstack just popped - plan not tentative	- use plan to generate equation - use previous equation to substitute for one of the Qknowns - isolate the Pfound - write the equation - add the Pfound to the Qknowns - add any other quantities to Qsoughts
(11)If - planstack empty - equation just formed - Qsoughts not in Qknowns	- lose confidence in the stack - replace Psoughts and Pknowns by Qsoughts and Qknowns

Notes on the Planstack Model

These notes refer to the eleven numbered procedures of the full planstack model.

(1) When everything that was originally sought is known, the problem has been finished. The result is then contained in the equations which can be solved to yield the quantities desired in the question. Quantities are added to the Qknown list only when an equation which solves for them has been generated.

(2) If the sketch test fails, it may be necessary to backtrack on this procedure, producing alternate sketches. If this is not possible, alternate parses. If this is not possible, the problem-solver will fail.

(3) This models the ability of experts to guess what principles will be involved before reading the whole question.

(4) It is not presently clear what tests would be applied. Hinsley, Hayes and Simon's example of the sawn board suggests that the information "A > B" cues a test to see if their instantiations obey this relation.

(5) This is the procedure that builds up the planstack. For experts, once it has been applied, it will be applied repeatedly until the planstack has created plans to solve for all the sought quantities.

(6) This covers the claim that experts can tell what quantities a plan will involve but novices can't.

(7) This is the "leaky planstack" assumption introduced to account for intermediate problem-solvers who appear to complete the construction of a planstack, but then find when popping it that it's incomplete, and doesn't provide a full solution.

(8) The stack is popped if it's complete. A novice pops the stack after each plan is constructed (and marked "tentative"), and an intermediate problem-solver pops the stack after every plan if the stack has leaked (and he has therefore lost confidence in it). The problem-solver loses confidence in a planstack if he generates an equation, leaving the planstack empty, yet does not produce an equation which solves for the original soughts.

(9) This procedure is used when the stack is popped after each plan. There is no substitution, because there will be no equation to substitute from. There is no isolation, because there will be no equation to substitute into. It is assumed that the equation can be solved to find the Pfound, so it is now added to the Qknowns.

(10) The expert procedure for generating an equation from a plan.

(11) This is the procedure that gets activated when a leaky stack is popped, and it is found that the plans in it are insufficient for a complete solution. The intermediate level problem-solver gives up the planstacking approach, and by losing confidence in the stack, ensures that subsequent equations are generated by forward inference.

The Hidden Curriculum Assumption

When people learn to solve problems, they are taught various explicit facts and skills. In addition, they infer things they are not explicitly taught, and which are not on the agreed curriculum. These may be called "Hidden Curriculum Assumptions".

One of them is the assumption that all the quantities in a question are relevant to its solution. Reliance on this assumption enables experts to reduce the number of principles they need to consider before selecting an appropriate one. From now on, this will be referred to as the "Hidden Curriculum Assumption of Non-Redundancy"

Another assumption which will be made use of when dealing with error generation and error modelling is the assumption that the novice problem-solver is expected to produce a certain kind of behaviour- in this domain he is expected to produce equations.

So if a novice is unable to solve the problems, or even generate a single plan, he has the alternatives of producing nothing, or a random equation. If we accept that it is more stressful to produce nothing than to produce erroneous output,

then we should expect the student to generate arbitrary results vaguely resembling equations that might result from the problem. This is essentially a random and syntactic process, yielding erroneous and unrepeatable results. From now on, this particular Hidden Curriculum Assumption will be referred to as the "Guess on failure" assumption.

Problem-Solving and Errors

The planstack model explains how people can solve problems, and what processes experts possess that enable them to exercise expertise.

It does not address the issue of how people learn to solve problems, how they learn the processes which result in expertise (aside from the plan compaction process), and it does not explain how people make errors.

Any analysis of novice scripts will confirm that errors form a large part of a novice's initial output. It is hard to believe that anyone could reach a state of expertise in the domain without having made any errors. Therefore we must look at

errors as an essential part of the learning process, and we must expect any comprehensive theory of how people learn such a skill to deal with errors as well as correct solutions.

In addition, there is a practical reason for studying the mechanisms underlying errors- if we want to teach people to do it correctly, we need to know why they do it wrongly. The overwhelming consensus of opinion in advanced CAL research, as dealt with earlier, is convinced that adequate remediation depends on accurate error analysis. Of course, the overwhelming consensus of opinion may be wrong, but we need a method of accurate error analysis before we can test such a hypothesis. This itself is another justification for studying the commission of errors.

All the models considered so far involve a stage at which the problem-solver has identified a principle and a context, and decides to use the principle to generate an equation. This is one place where errors might occur (and do occur).

Are there any others? The novice might of course parse the problem wrongly, or sketch the problem wrongly. We shall not consider these processes further in this paper. He might solve the equations wrongly (but we shan't deal with this either). But the internal nature of the planning carried out by the

planstack model implies than an error in processing between the sketch and equation generation stages would not result in an overt error. It would result in a null output, or in the activation of the "Guess on failure" assumption.

Thus the implications of the model are that all structured and repeatable errors taking place in correctly parsed and sketched problem occur at the "generate equation" stage. Since this stage is not exclusive to the planstack model, it is possible to investigate the occurrence and genesis of errors independently of the control strategy used by the problem-solver.

Larkin has adapted the ABLE model to duplicate student errors. The ABLE model very naturally accounts for "limited-force bugs", in which one or more forces are omitted from a resolution equation. In addition, the model was adapted to individual students' behaviour by including special-purpose productions.

As Larkin says:

> "In a few cases, a subject simply wrote an incorrect equation. In that case, the corresponding production in his version of slightly ABLE was modified to write that incorrect equation. Generally, a subject's work for the two problems was sufficiently consistent that productions acting in one problem did not have to be modified or deleted to account for work in the others."

Larkin did not describe the errors (other than limited-force bugs), nor indicate whether they were repeated in different problems. Her subjects were relatively skilled novices solving dynamics problems.

The remainder of this paper deals with the identification of errors in the solution of statics problems. This is done by analysing novice scripts, and comparing them to the output of a computer model called NEWT, which solves problems in the domain both correctly and erroneously.

Chapter V - Methodology

This Chapter considers how computational models can be evaluated by comparison with experimental data, and defines a method for doing so which will be used to evaluate the NEWT system.

Structure of the Psychological Theory

The psychological theory presented in this thesis consists of three parts:

(1) A theory of competence

(2) A theory of consistent errors

(3) A theory of inconsistent errors

Of these, (1) and (2) are Type I theories in Marr's Typology, (Marr, 1977), while (3) is not a theory that generates behaviour, and hence does not count as an AI theory as such (there is no algorithm for modelling inconsistent behaviour).

The theory of competence deals with how problem-solvers solve problems correctly, but does not address the question of why they solve some problems and not others- it is a theory of problem solution, not problem choice.

The theory of consistent errors deals with what problem-solvers do when they make consistent errors. It does not deal with the question of why problems that cannot be solved are attempted.

The theory of inconsistent errors deals with the question of what problem-solvers do when they attempt problems which they cannot solve consistently (either correctly or incorrectly). It does not deal with the question of why such questions are attempted.

None of these three theories deals with the question of what happens when a problem-solver does not attempt a problem.

The Theory of Competence

This deals with how problem-solvers solve problems correctly. Its implementation is the MECHO program of Bundy et al, and is presented as a theory of novice problem-solving rather than expert problem-solving.

This theory produces output which is matched to that of the subject investigated at the level of equation generation. It claims to predict the equations that will be generated, and the intermediate quantities that will be introduced by competent novice problem-solvers.

The issue of equation ordering has been extensively dealt with in the previous chapter, and is not considered further in this study. Results supporting the claims of MECHO to duplicate human problem-solving behaviour in equation generation and quantity introduction have been presented by Luger. These claims will be looked at in more detail later.

The full implementation of MECHO deals with a variety of physical laws and problem types, but the problems investigated in this study were drawn exclusively from the field of statical

equilibrium. This theory predicts consistent correct solutions.

The Theory of Consistent Errors

This claims that most consistent errors are produced by of one of a limited number of incorrect solution procedures. These incorrect procedures, or 'malrules', resemble correct procedures for generating equations from physical principles; and are applied consistently by the problem-solver whenever the principle is used, instead of the correct procedure.

The malrules hypothesized to exist are listed later on. Choice of principle is as for the theory of competence. This theory predicts consistent incorrect solutions.

The Theory of Inconsistent Errors

The correct procedures and the malrules include nearly all the consistent procedures that can be applied to yield equations. Problems-solvers who do not possess either a correct or an incorrect procedure relevant to the context are hypothesized to produce random equations using the quantities mentioned in the question. Such behaviour was also proposed by Scanlon, Hawkridge and O'Shea (1983) in similar circumstances.

Being random, this behaviour is inconsistent, and if the problem-solver attempts the same question again, there is no reason to expect him to produce the same results. Generally, problem-solvers realise they are acting randomly, and when asked to repeat the question, they produce no output at all.

Presumably the reasons why people attempt questions they know they cannot solve is to be found in their social expectations- the theory does not deal with this issue.

The predictions of this theory are that if, on the first attempt, a problem-solver does not use correct or malrule procedures, then the problem-solver will not produce the same result if asked it again.

This theory does not attempt to predict or model the random output hypothesized as the behaviour when dealing with a question for the first time. There is consequently no computational implementation of this theory.

Derivation of the Malrules

The malrules implemented in NEWT were identified from problem sheets answered by 28 students in 1981. The students were aged 16-18, and were studying either O-Level physics or TEC Level 1. This set of scripts will be referred to from now on as the "training set".

Subsequently, 23 other students were given problem sheets, and their solutions were compared to those obtainable from NEWT. This second group of students will be referred to as the "validation set", and the results obtained will be considered in detail in Chapter VII.

The Theories and the NEWT program

As we have seen, there is no consensus as to how to relate a theory to a program. The solution attempted in Ch VI will be:-

(a) to present the theory of competence, in natural language

(b) to present the top-level code in the NEWT program corresponding to the control algorithm

(c) to comment on the code presented in order to relate it to the theory of competence

(d) to list the malrules used by NEWT in natural language only. The theory of consistent errors consists of the theory of competence together with the available malrules.

As these are presented as Type I theories, it is the relationship between the natural language theory and the data which is the vital criterion of validity: the details of implementation are relevant only in as far as they do or do not produce the behaviour called for by the theory.

Of course, in a large program such as NEWT, it has to be taken on trust that there are no procedures that invalidate the claimed correspondence between the theory and the code. In the last resort their non-existence can only be proved by having the program independently rebuilt from its description.

Relating the Model to the Data

The question will now be dealt with as to how the model is evaluated with regard to the data.

In this study, the aim is to construct a "student model" for each student who makes consistent errors. The theory of competence is itself the student model of correct novice problem-solving, and the model for each error-making student thus consists of the competence model peturbed by the replacement of the correct equation generation procedure by one or more malrules selected from the pool of available malrules.

The set of data points over which the model is kept constant will be termed the "span of consistency"; and the set of data points over which the data is compared to the output of the model will be referred to as the "span of comparison". These

will differ between the three different psychological theories, and also between them and the evaluation of NEWT as a student model, as will be explained later.

Clearly, different error-making students will in general be represented by different models. Once a model has been selected for a particular student (in the study, the appropriate malrules were selected by hand), it is necessary to come up with some measure to evaluate the model.

Because there is no alternative hypothesis- no theory of what the errors would be if the proposed theory were not true- the normally used tests of statistical significance cannot be applied to such a measure. Nor is there a well defined measure to be suggested by statistical theory. As Newell and Simon say in "Human Problem Solving":

> "There is not even a well behaved euclidean space of numerical measurements in which to plot and compare human behaviour with a theory. Thus, this book makes very little use of the standard statistical apparatus."

Another reason for the inapplicability of conventional statistical procedures is the fact that the models are constructed after the data has been collected, and are

deliberately intended to maximise correspondence with that data.

Newell and Simon's comment remains true today, and it therefore becomes necessary to select an arbitrary measure of evaluation. Of course, the "standard statistical apparatus", itself makes use of arbitrary conventions in choosing particular levels of significance at which to accept or reject a hypothesis.

There is no reason to suppose that any evaluation algorithm will yield an unambiguous answer to the question "Is the theory a good one?" without referring at some point to an arbitrary value of what counts as 'good'!

Theories of the type described above are described by Newell and Simon, who used the term "IPS", or "Information Processing System", in the sense that I have used the word "model":

"... the theory of human problem-solving as set out in this book produces a large number of highly specific little theories- microtheories would be an appropriate term. Each of these microtheories can be viewed as a single data point, at which a more generalised theory is put to the test. Each data point requires this elaborate treatment because the data with which it makes contact are so sequentially independent that

they cannot be unravelled meaningfully without the creation of a highly specific IPS".

From now on, a psychological theory which requires to be "tuned up", to produce a micro-theory for every subject, will be referred to as a "micro-theory system".

What Newell and Simon refer to as "the standard statistical apparatus", is concerned with the answer to the question "How sure are you that the predicted results don't correspond to the data by chance?"

This is not a relevant question in this domain : the probability of a monkey at a typewriter coming up with the same equation as a problem-solver is so remote as to be negligible.

The question that needs to be answered is rather "How sure are you that the predicted results don't correspond to the data in a trivial manner?"

Since the theory may incorporate arbitrarily many procedures of arbritrarily great complexity, it is possible to posit a separate micro-theory for every data point. Such a micro-theory system necessarily accounts for all the target data, but in a trivial way that vitiates its explanatory power.

Any proposed measure of evaluation needs to exclude such theories in an objectively definable manner. From here on, we shall assume that each micro-theory consists of a core procedure, together with none or more variant procedures. In the NEWT system, the core procedure corresponds to MECHO, while the variant procedures are the implementations of the various malrules for the domain.

Criteria for an Evaluative Measure

Six criteria for an adequate evaluative measure for a micro-theory system are here proposed, as described by Priest and Young (1986):

(1) The Numeric Value Criterion

The measure consists of a single numeric value. This ensures that measures for different theories are well-ordered.

(2) The Accuracy Criterion

The measure should increase every time the model matches the data.

(3) The Inaccuracy Criterion

The measure decreases every time the model fails to match the data.

(4) The Parsimony Criterion

The measure should decrease for every time a new variant procedure is introduced to explain the data.

(5) The Prediction Criterion

The value of the measure over a given data sample should provide an unbiassed predictor of its value over a larger sample.

(6) The Assumption Criterion

The measure should not involve the assumption of any particular distribution of micro-theories- for instance, that the micro-theories corresponding to common behaviour are discovered before micro-theories corresponding to rare behaviour.

Evaluation of Micro-Theory Systems

Some micro-theory systems are easier to evaluate than others. One outstanding regularity of the domain under consideration is that it consists of independent units of behaviour, (in this case equations), each of which can either be fitted by the data (a 'hit') or not (a 'miss'). So a measure calculated on the basis of a single count of 'hits' and 'misses' becomes possible. Such a measure would not take any consideration of how "far" a miss was from a hit, and would regard all hits as equally important.

It would not take account of any relationship between hits and misses- this would be done by the model or not at all.

The Error Fit Measure

A micro-theory system such as this was produced by Young and O'Shea (1981), who considered the issue of evaluation. They produced a model of human subtraction problem-solving which combined correct procedures together in such a way that rule

deletion produced many of the observed errors. Thus their theory could not be divided into a theory of competence and a theory of errors, and one measure had to cover the adequacy of the theory for both correct and incorrect solutions.

Output was matched to the data at the level of the numerical answer. Thus the model produced the same answer as the subject (a hit), or a different answer (a miss). Their chosen evaluation measure was described thus:

> "Our evaluation technique is to subtract the false errors from the hits, i.e.: to calculate:
>
> net score = (hits - false errors)"

In this calculation "false errors", are errors predicted by the model on questions which the student solved correctly. The span of consistency for their model was the set of questions attempted in a single script, and therefore it was possible for their model to predict errors on questions where no error occurred. They did not take questions that were done correctly, and which were predicted by the model to be done correctly, as affecting the "net score".

The net score was used by Young and O'Shea by comparing it with the number of errors in the data.

> "Comparing the net score of 128 with the 178 errors provides the answer to the question of how well the PS accounts for the data. The answer is that the PS accounts for a little over two thirds of the errors".

Strictly speaking, this conclusion only follows if the net score of 128 is made up of 128 hits and no false errors. If the score were the result of 178 hits and 50 false errors, then the model ("PS" in their terminology) would of course account for all the errors, not just two thirds of them. The figure of two thirds relates to the smallest possible proportion of errors accounted for.

Thus by implication, Young and O'Shea are making use of the measure:

$$\frac{\text{Hits - false errors}}{\text{errors observed}}$$

This is referred to as the "Error Fit Measure", and satisfies criteria (1), (2), (3), (5) and (6) given above, but not (4). The presentation of their results by Young and O'Shea implies

that they consider a value of 2/3 as a "good fit", an implication which has not been disputed in the literature. Their defined measure does not take account of the number of variant procedures used in the study, although they were extremely parsimonious about their introduction.

The Micro-Theory Evaluation Quotient

In order to evaluate the micro-theories proposed to model individual students by a measure that satisfies criteria (1), (2), (3), (4) and (6) proposed above, I define the Microtheory Evaluation Quotient, or mu-quotient, of a micro-theory system as follows:

(a) The theory is adapted to form micro-theories over the appropriate span of consistency. No variant procedure is introduced unless it accounts for at least one hit.

(b) The micro-theories are compared to the data over the appropriate span of comparison. This must include any false errors for the particular micro-theory.

(c) Each datum is compared to the corresponding output of the relevant micro-theory, whose output is classified as a "hit" or a "miss".

(d) The mu-quotient of the theory is defined as:

$$\text{mu} = \frac{\text{hits - variant procedures}}{\text{data items}}$$

There are various points to be noted about this seemingly simple definition:

- the mu-quotient is a measure of the microtheory system, not any particular micro-theory

- the span of consistency and the span of comparison need to be chosen carefully with respect to the particular theory. This point will be considered below.

- it is not necessary to include a count of 'false errors', in order to satisfy the proposed criteria. If the span of comparison is chosen so that such events count as data items, then an increase in the number of false errors increases the denominator of the quotient, and hence reduces the overall value of the mu-quotient.

- the mu-quotient will be a dimensionless number in the range $0 <= mu < 1$. Since the introduction of a micro-theory only takes place if it results in a hit, the minimum value of the numerator is zero. The measure never reaches a value of unity however, but only approaches it asymptotically as the number of hits by the same set of micro-theories increases.

- The number of micro-theories looks at first like the number of degrees of freedom in other forms of statistical analysis. The two are not the same because a micro-theory has an internal structure- it is not a numeric variable. The measure proposed does not depend on the internal structure of the micro-theories used.

- The criterion not met by this measure is that of Prediction. Using this measure on a small sample necessarily yields a lower value for mu than would be expected from a large sample.

The Span of Consistency for NEWT

For the theory of competence and the theory of inconsistent errors, no alteration is required in the micro-theory system when moving from one student to another. The span of consistency is therefore the entire set of data points.

For the theory of consistent errors and the student model, the appropriate span of consistency depends on how consistent the students themselves are in their solutions. If students are rigorously consistent in the errors they commit, the appropriate span of consistency would be all the solutions of a single subject. However, if students were not consistent between questions, but were consistent in dealing with repeats of the same question, the appropriate span of consistency would be a single repeated question.

In advance, it is not possible to tell how consistent subjects will be, and therefore analyses at both spans of consistency will be performed. One of the results to be obtained should therefore be a measure of the consistency of the subjects in their problem solutions.

The Span of Comparison for NEWT

When it comes to considering the span of comparison for NEWT, there are four separate theories to consider. Each of these operates over a different span of comparison, and consequently results in a different mu-quotient.

In the experimental study described below, each question was presented twice. It was therefore possible to distinguish consistent behaviour by the subjects, in which the same question resulted in the same behaviour, from inconsistent behaviour, in which the subject's behaviour was different on the different attempts at the same question.

The four theories to be considered are:

1) A theory of competence.

This predicts how people will solve questions that they solve correctly. The span of comparison for this theory is therefore the set of questions solved correctly on the first attempt.

2) A theory of consistent errors.

This predicts how people will solve problems that are solved incorrectly but consistently. At first sight it might seem that the span of comparison should be the set of incorrect but consistent questions. This does not allow for "false errors", however. In order to reduce the mu-quotient for every error predicted where the subject in fact proceeds correctly, it is necessary to define the span of comparison to consist of the set of questions consistently answered for which the model predicts an error.

3) A theory of inconsistent errors.

This predicts that people who do not make errors corresponding to malrules in the theory when attempting a question the first time, will produce inconsistent results when given the same question again. It may be reformulated as that the theory covers all possible consistent errors, and that any errors not covered are not consistent. The span of comparison consists of all those error questions which cannot be assigned a model by the system.

4) A student model for predicting behaviour.

This is not a psychological theory as such, and so different mechanisms for producing different types of behaviour do not need to be differentiated.
The span of comparison consists of all those questions attempted the first time.

It was felt necessary to test the theory of competence and the theory of consistent errors separately, because otherwise an arbitrarily high mu-quotient could be obtained by including scripts from competent students who solved problems correctly. It is obvious that such data should not affect the validity of a theory of errors.

Summary of Evaluation Criteria

Theory	Consistency	Comparison	Prediction
Competence	single subject	all questions answered correctly the first time	consistent behaviour
Consistent errors	single subject	all questions answered consistently and observed or expected to contain errors	consistent behaviour
Consistent errors	single data item	as above	as above
Inconsistent errors	-	all questions that NEWT is unable to model	inconsistent behaviour
Student model	single subject	all questions attempted the first time	consistent or null behaviour
Student model	single data item	as above	as above

Repeated Answers

Sometimes, a student answered a repeated question by repeating the answer they gave on the first occasion. In the absence of any theory to account for why some questions were answered by solving again from scratch, and some by simply repeating the previous answer; this was arbitrarily designated as "consistent behaviour".

Matching Output and Data

The evaluation described above depends on being able to match the output of the model to the data by a method that classifies the output as a "hit", or a "miss", for each data item in the span of comparison.

In this study, a "data item", is the student's attempt at solving a pair of identical questions and the matching between the output of the model and the data is at the level of equations generated.

In order to identify "hits", and "misses", a number of conventions are required to categorize the output in those cases where there is not a direct symbol-to-symbol congruence between the data and the output.

(1) Only equations resulting from the instantiation of a physical principle are counted. This is because NEWT is a theory of physics problem-solving, not algebraic competence.

(2) Two equations match if one can be derived from the other.

(3) Two equations match if one can be derived from the other after a substitution of numerical values from the question.

(4) Only the equations produced in the data are targets for matching. Since NEWT does not incorporate a theory of halting, subsequent equations produced by the model are considered irrelevant when deciding if the output matches or not.

(5) Equations containing structured numeric data are considered as targets for matching, but not equations consisting of single number equated to a quantity.

Thus: X = 30 * 2 + 5 is a condidate for matching

but: X = 65 is not

This is because it is impossible to tell from the number alone what process led to its calculation. This convention does not hold for repeated questions. Here, a remembered answer counts as equivalent to the original working.

(6) If no sought variable occurs in an equation, it is considered as "rough working", and not considered as a target for matching.

(7) Lines of working not having the correct syntactic form of an equation are not considered as targets for matching.

Other forms of Evaluation of Theories

Van Lehn (1981), does not believe that relating a theory to a program and testing the program against the data is sufficient to validate the hypothesis. He feels that in addition, the technique of "competitive argumentation", should be used; and himself illustrates its use in considering the mechanisms underlying Repair Theory. As he says:

> "Matching hypotheses is only a part of developing a theory. The other part is validating those hypotheses. An important validation technique used with this theory is competitive argumentation... One shows that a hypothesis accounts for certain facts, and that certain alterations to the hypothesis, while not perhaps without empirical merit, are flawed in some way. That is, the argument shows that its hypothesis stands at the top of a mountain of evidence, then proceeds to knock the competitors down".

In cases where there are alternative hypotheses proposed to account for the data, this seems appropriate. In the domain covered by NEWT, however, there are no alternative proposed

theories, nor am I able to think of any. So there will be no competitive argumentation in this thesis.

Student Models in Intelligent Tutorial Systems

Sleeman and Brown (1982), identify three major problems with the use of student models in intelligent tutorial systems. These are:

(1) The Assignment of Credit

(2) The Combinatorial Explosion of Compound Concepts

(3) Discounting Noisy Data

The likely effect of these on the implementation of a student model based on NEWT will now be considered.

The Assignment of Credit problem occurs when the student halts, and does not produce output. To what mechanism or malrule should the "Credit", for this be assigned in the student model?

NEWT is unable to provide any solution to this, since it does

not embody a theory of halting. It is sometimes able to predict halting, but this does not enable it to "assign credit", to any student model mechanism, or to come to any conclusion about the student other than that he is unable to apply a particular physical principle at all. Whether this is sufficient information for a tutorial strategy to be able to work effectively must remain an empirical question.

The Combinatorial Explosion of Compound Concepts is a practical problem where students produce output to which a large number of possible malrules may apply. This is also a difficulty for a NEWT- based system in principle, but in practice, multiple malrule behaviour is relatively uncommon in this domain. Furthermore, many of the possible malrules are incompatible, and so cannot be applied simultaneously.

As well as these ameliorating circumstances, the difficulties introduced by the possibility of multiple malrules are reduced by the fact that many malrules affect only certain terms of the output, and can be identified from the particular term they would affect, independently of other malrules.

Taken together, these facts suggest that combinational explosion of multiple malrules is not likely to be an insuperable problem in practice.

Discounting noisy data is a process partly accomplished by the concentration on equations as the points of contact between the theory and the data.

If the input processor of a teaching program were to incorporate some method of constraining the student's input to equations whose syntactical correctness could be checked, this would allow noisy data to be reduced to a level comparable to that found in the data analysis that follows. This would be a non-trivial task, but appears to present no insuperable difficulties in principle.

Conclusions

This chapter has considered two different approaches to the relationship between a psychological theory and a computational model, corresponding to Marr's Type I and Type II theories.

There are four separate theories to be considered and evaluated in the rest of this thesis:

(a) A Type I theory of competent problem-solving corresponding to the MECHO program.

(b) A Type I theory of consistently erroneous problem-solving corresponding to the NEWT program.

(c) A theory of inconsistent problem-solving which is not expressed computationally.

(d) A Type II theory covering the whole range of problem-solving behaviour, intended as a basis for an intelligent student model. This also corresponds to the NEWT program, viewed over a different span of

comparison from the theory of consistently erroneous problem-solving.

Methods of evaluating computational models have been considered. The critical question to be addressed has been identified as how one can demonstrate that the program does not achieve its results trivially.

Six criteria for an ideal measure of the parsimony of a computational model have been proposed; and a measure has been defined which satisfies five of the criteria. This measure, the micro-theory evaluation quotient, will be used to evaluate the four theories in Chapter VII.

Chapter VI - Design of the NEWT System

The NEWT system may be considered in two parts:

(1) The selection of physical principles and contexts in which to apply them.

(2) The generation of equations when given physical principles and contexts.

The mechanism corresponding to the first part derives from the MECHO program, and implements the major part of the framework for competent problem-solving.

The mechanism corresponding to the second part contains the correct equation generation procedures from MECHO, and also procedures corresponding to every identified malrule.

The set of procedures corresponding to malrules defines the theory of consistent errors and by extension defines the theory of inconsistent errors also.

Selection of Principles and Contexts

When the problem representation has been read into the database, and NEWT has started solving the problem, it first identifies lists of 'sought' and 'given' quantities. It then applies the Marples Algorithm (Marples, 1974), to generate successive plans for producing equations. Each plan contains a physical principle and a context in the problem representation.

Plans are generated as follows:-

(a) Take the first sought quantity.

(b) From the problem description, identify the type of the sought quantity.

(c) Choose a principle which relates a quantity of this type to other quantities. The only principle available to NEWT in the Statics domain is the Resolution of Forces Principle. The MECHO program is able to use a wider variety of principles, such as the Principle of Moments, various kinematic principles, and so on.

(d) Choose a context from the problem representation in which to apply the principle. For the Resolution of Forces principle, the context must be a point or a rigid body, together with a direction. The body or point is chosen from amongst those related to the sought quantity, and the direction will be that of one of the forces acting on the point or body.

A list of plans already used is kept, and it is forbidden to choose a principle and context that have been chosen before. This guarantees that the equations generated will be independent, and that a solution can therefore be found.

(e) Control is now passed to the mechanism for generating equations from plans. There are two of these:

- a mechanism to generate equations which does not allow itself to introduce new quantities.

- a mechanism to generate equations which does allow itself to introduce new quantities.

The control strategy is to use the first equation generating mechanism if possible, and the second mechanism if the first one fails.

(f) If an equation containing the sought is generated, that quantity is removed from the list of sought quantities and added to the list of known quantities. The system then recurs from (a) until no sought quantities remain.

(g) If no equation containing the sought quantity is generated by step (e), the system backtracks to (d) to choose an alternative context, or to (c) to choose an alternative principle if no alternative context can be found. If no alternative choice of principle is possible, the problem-solver fails.

These steps are common to the theory of correct problem-solving and the theory of consistent error generation. Errors in this basic control structure do not result in the generation of erroneous output, but in non-optimal solution paths or premature halting behaviour.

Generation of Equations

For every allowable principle and type of context, NEWT maintains a correct procedure for generating an equation. For the Resolution of Forces principle, NEWT also maintains a procedure for generating an equation corresponding to each malrule identified. Thus a particular micro-theory consists of the plan selection mechanism, together with a subset of the equation generation procedures.

Each procedure can behave in one of two ways- either it can be allowed to introduce new quantities or it can be prevented from doing so. This choice is made by the plan selection mechanism when the procedure is called.

Equations are generated from the Resolution of Forces Principle as follows:-

(a) All known tension, compression and applied force components acting on the object in the direction of resolution, are collected.

(b) The component of the weight acting on the object, either known or inferred from the mass, is added to the collection of force components.

(c) If the object is in contact with another object or surface, a force of reaction in accordance with Newton's third law is inferred and added to the force component collection.

(d) If the object is in non-fixed contact with a rough surface, a force of friction, either limiting or non-limiting as required, is calculated and included in the force component collection.

(e) The force components collected are summed and equated to zero.

The final stage of equation generation is algebraic summation. Hence the final equation will contain a sum of terms, each of which represents the component of a force.

Malrules

There are five categories of malrule implemented in NEWT. Two categories are general malrules which may affect more than one term in an equation, and three are force-specific malrules which only affect terms relating to forces of a specific type. Twenty-four malrules in all have been induced from the training set. The categorisation into five groups has been chosen as it appears to correspond to the different procedural mechanisms involved in the operation of the malrules.

I Malrules of Weight (3 malrules)

II Malrules of Reaction (1 malrule)

III Malrules of Friction (4 malrules)

IV Malrules of Component Calculation (12 malrules)

V Malrules of Vector Addition (4 malrules)

Each malrule has been numbered for ease of reference, but these index numbers are not intended to possess any significance. The twenty-four malrules identified in the study are:

Malrules of Weight:

1. Weight = mass

2. Weight = mass + 9.8

3. Weight = mass * 9.8 * 10

Malrules of Reaction:

4. Reaction = Weight

Malrules of Friction:

5. Friction = mu * Reaction (when the friction force is non-limiting)

6. Friction = 0 (when the friction force is limiting)

7. Friction = Reaction

8. Omit the vertical component of friction.

Malrules of Component Calculation

When a force R is resolved parallel to a line inclined at an angle of a to its line of action, the magnitude of the resolved component C should be R * cos a.

9. C = R - cos a

10. C = a - cos a

11. C = R * cos theta (where theta does not appear in the problem representation)

12. C = R * a

13. C = R + sin a

14. C = R / tan a

15. C = cos a

16. C = R * sign(cos a) where sign(x) = 1 if x > 0
= 0 if x = 0
=-1 if x < 0

17. C = R * sin a

18. C = R / a

19. C = cos theta

20. C = sin theta

Malrules of Vector Addition

21. When friction, weight and reaction are in equilibrium:

Friction = Weight - Reaction

22. When a body lies at rest on a rough slope:

    Friction = 9.8 - Mass

23: If three forces are in equilibrium at a point and one is known, each of the others are equal to it in magnitude.

24. When a known and an unknown force are in equilibrium together with a third force normal to the direction of resolution,and the angle between the unknown force and the direction of resolution is a:

    Unknown Force = Known Force * cos a

For reasons of space and clarity, the malrules are only presented here in natural language form. The categorisation of malrules into five categories is based on the assumption of the existence in human problem-solvers of at least five distinct mechanisms:

- one to identify the weight

- one to identify the reaction

- one to identify the friction

- one to calculate the component of a force at a given angle

- one to add vector quantities

The last two classes of malrule are mutually exclusive- either the problem-solver attempts to take components of all relevant forces and equate the sum to zero; or they use a "vector addition" method which does not rely on taking components.

Malrules in Operation

This section will look at the operation of malrules in practice, and consider some of the possible mechanisms by which they might work.

Malrules of Weight: The first of these would appear to derive from a confusion between the concepts of mass and weight, while

subjects evincing the second and third malrules clearly know that there is a difference between the concepts. They also know that mass and weight are related by a constant, but have not learned what the correct relationship is. One may also infer that the users of the second malrule do not recognise that mass and weight are measured in different units- they are simply mis-remembering an algorithmic procedure.

Malrules of Reaction: Only one malrule has been included in this category. This does not mean that there is only one kind of error that can involve a force of reaction- malrules of components or vector addition may result in errors involving a reaction term as well. This malrule is an oversimplification of the correct definition of reaction- in the case where a body rests on a horizontal surface, it yields the correct answer.

Malrules of Friction: The first of the friction malrules is an overgeneralisation of the friction law to cover cases of non-limiting friction. The second malrule involves the omission of limiting friction altogether. This might conceivably be due to an inability to deal with a number of forces simultaneously, leading to the omission of the least obvious force. Alternatively, this might result from a process of naive physical reasoning of the form "Friction is what tries to stop a body from moving: it hasn't been stopped from moving, so

there is no friction". The malrule "Friction = Reaction", is apparently due to the incomplete recollection of the fact that the friction between a body and a surface depends on the normal reaction between them. The "Omit Vertical Component of Friction" is again an oversimplification, presumably generated from the recollection of examples where bodies were placed on horizontal planes, and friction did not have a vertical component.

Malrules of Component Calculation: Nine of these are variations on the theme that the magnitude of the component depends in some way on the magnitude of the force and its orientation. It is doubtful whether all of these are malrules in the sense of being relatively permanent structures- many will probably have been created as temporary patches and never used again. The remaining three malrules (Nos 11, 19, and 20), have the appearance more of "formulae" than of equations. From the context in which they were observed, it is clear that they were not intended as formulae, because they occurred within partially instantiated equations containing quantities mentioned in the problem description. There would seem to be a deep confusion between formulae and equations in the minds of the subjects who produced these errors.

The distinction between a formula in which each symbol stands for a true variable, and an equation in which each symbol stands for a (possibly unknown) but definite value; is perhaps blurred in the minds of some inexperienced students by the archaic and educationally reprehensible practice of using syntactically similiar symbolic names for both cases. Thus novices may be introduced to the constant acceleration kinematic formula " v = u + a * t ", in which the symbol "v" stands for the final velocity of any object at the end of any interval. They may then go on to apply this formula to a question in which the symbol "v" is used to stand for the velocity of a particular object at the end of a particular interval. No wonder some students become confused between the two concepts.

Malrules of Vector Addition: These are the most complex of the malrules, and are applied as an alternative to the summation of force components. The first of these is obscure in its derivation- all that can be said with any certainty is that the user does not understand the difference between vector quantities and scalar quantities. The second could conceivably be composed from two separate malrules:

Friction = Weight

Weight = 9.8 - Mass

There is no independent indication of the existence of these rules, and hence this is counted as a single rule. The fourth vector addition malrule is an oversimplification which yields a correct equation in the case when the three forces are inclined to each other at equal angles. The commonest malrule in this category is the last one. The correct equation to produce would be of the form:

Unknown Force * cos a = Known Force

which bears a superficial resemblance to the malrule version, but is harder to solve.

Examples of Malrules

There follow examples of the equations generated by the twenty-four malrules, as applied to either the Strut Problem (Fig 1), the Slope Problem (Fig 3), or the Angle of Friction Problem (Fig 4).

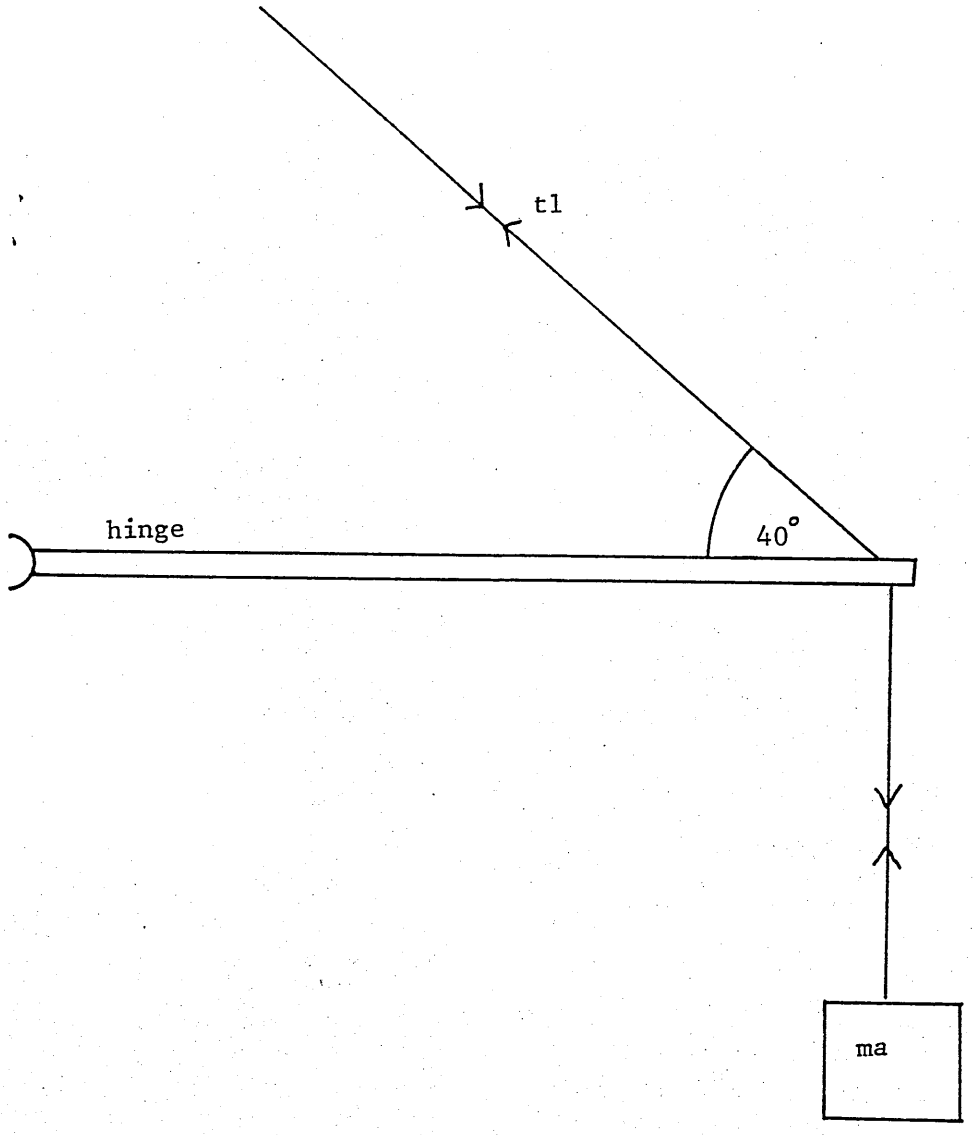


Figure 1. The Strut Problem

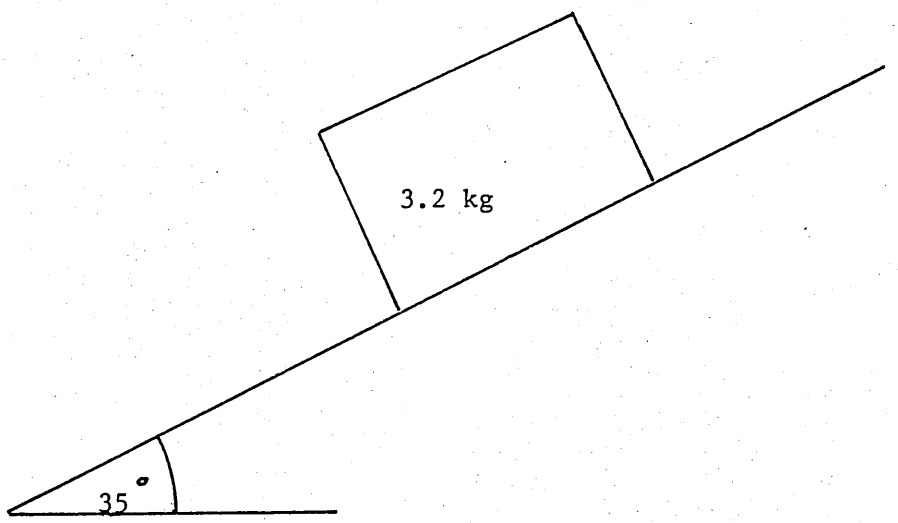


Figure 3 . The Slope Problem

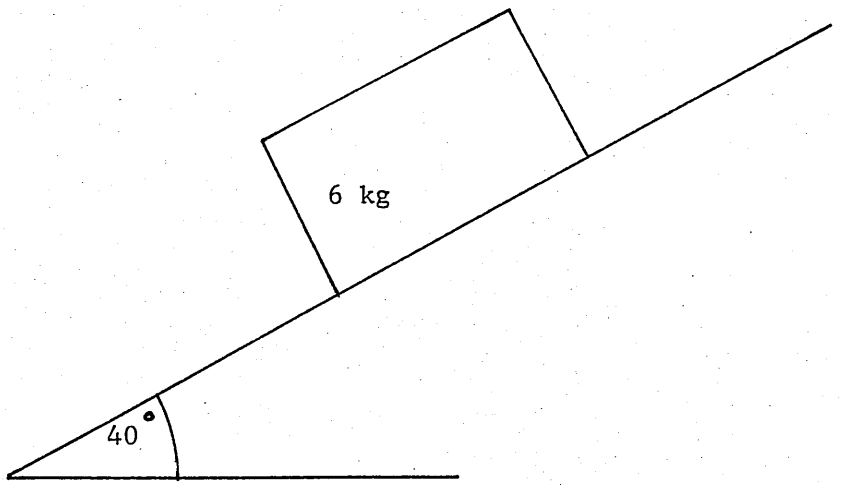


Figure 4. The Angle of Friction Problem

Malrule1 (Strut Problem):

t2 = ma

Malrule2 (Strut Problem):

t2 = ma + 9.8

Malrule3 (Strut Problem):

t2 = ma * 9.8 * 10

Malrule4 (Slope Problem):

reaction1 = 3.2 * g

Malrule5 (Slope Problem):

friction1 = mu * 3.2 * g cos 35

Malrule6 (Angle of Friction Problem):

6 * g = reaction1 * cos 40

(where reaction1 is the force of reaction between the block and the plane)

Malrule7 (Angle of Friction Problem):

6 * g = reaction1 * sin 40 + reaction1 * cos 40

Malrule8 (Angle of Friction Problem):

6 * g = reaction1 * cos 40

(Notice that in this example, the malrule yields the same results as Malrule6. The two malrules can be distinguished by the equations generated with other questions.)

Malrule9...Malrule20

These involve a variety of bizarre ways of calculating the components of a force. Two of these will be selected as examples.

Malrule9 (Strut Problem):

t1 - cos 40 = t2

Malrule11 (Slope Problem):

reaction1 = 3.2 * g * cos theta

Malrule21 (Slope Problem):

friction1 = 3.2 * g - reaction1

Malrule22 (Slope Problem):

friction1 = 9.8 - 3.2

Malrule23 (Strut Problem):

t1 = t2

Malrule24 (Strut Problem):

t1 = t2 * cos 40

Relationship between MECHO and NEWT

The analysis of student errors presented in this thesis makes use of the MECHO program (Bundy et al 1979) to provide a basis of problem-solving techniques. This system is capable of solving problems in the domain under consideration here, as well as in other domains.

The NEWT program which will be compared below to the student output, consists of the MECHO program supplemented by additional procedures corresponding to executable versions of the malrules described in this chapter.

Conclusion

This Chapter has described the NEWT system in detail. The control strategy has been described (and is related to the code by which it is implemented in Appendix I). The twenty-four malrules inferred from the training set have been defined and illustrated. In the next chapter, their output will be compared to the equations generated by the students in the validation set.

Chapter VII - The Experiment

The results of an experimental study are presented in detail. This data is compared to the output of the NEWT system. Evaluation measures are presented for each of the three psychological frameworks, and for NEWT as a student model. In this study, the malrules inferred from the training set, and implemented as procedures in NEWT, were compared to the solutions of the students in the validation set.

The Subjects

Twenty-three students, here referred to as S1..S23 answered a test containing questions on statical equilibrium. Fourteen of the subjects were in a class studying A-level Physics, and eight were in a class studying A-level Applied Mathematics. The ages of the group ranged from 17 to 19.

The first test contained four questions and when this had been completed, the students were immediately asked to take another test. This contained the same questions as the first test, but

in a different order. One hour was allowed for the completion of both tests.

NEWT was adapted to model each student by including any malrules necessary to duplicate the student's first set of solutions as closely as possible. The questions in the tests are those given as examples in Chapter II, and the experimental data is presented in Appendix I of this chapter.

The four questions used in the experiment involve calculating:

-limiting friction

-non-limiting friction

-weight

-reaction

-tension

-multi-step solution paths

-components of forces at various angles.

The questions therefore cover a significant part of the domain. Four questions were considered sufficient because it was felt that for a student to complete four questions twice was as much as could be expected in one session. Taking experimental data from more than one session was considered, but was rejected since it would introduce the possibility that students might learn between sessions, thereby affecting the results.

Another reason for avoiding extra data-collection sessions with the same students is the "bug migration" phenomenon (Brown and VanLehn 1980). This involves the presence of a consistent bug exhibited over a period of time, which changes to a different consistent bug at a later period. In order to avoid the problems of learning and bug migration, data collection was restricted to a single session, and therefore to a set of problems that could be done in this time. Four problems seemed to be as many as most of the students could cope with in the time available. There is no statistical procedure for calculating a minimum neccessary number of questions in this case; but the experiment may be compared with other data collection exercises intended to analyse the methods used by experimental subjects to solve physics problems.

Thus Luger (1977) used three questions to extract protocols from three subjects, which were compared to traces

from the MECHO program. Larkin (1980) used five questions with her subjects (of whom there appeared to be eleven). Larkin, McDermott, Simon and Simon (1979, 1980) carried out two studies. In one, two subjects solved eighteen problems; and in the other, twenty-two subjects solved two problems.

Thus the number of questions used in this study is broadly in line with the numbers used by other workers in this field.

Observations on the Data

It may be observed from the data summarised below, that many students, unlike NEWT, use the friction law as an independent principle to generate separate equations while others, however, incorporate this relationship into the relevant term of the Resolution of Forces principle. From the point of view of identifying erroneous equation generating procedures, we shall treat the equation F = mu * R as if it were used to substitute for F in a Resolution equation.

e.g. the conjunction of equations

F = T F = mu * R

will be treated as equivalent to the equation

$$T = mu * R$$

Another observation that may be made is that several students solved problems by means of Lami's Theorem. This was not provided for by MECHO, since MECHO's existing methods were sufficient to solve any problem on which Lami's Theorem can be used. None of the students in the training set made use of Lami's Theorem, and hence it was not incorporated into NEWT. This reduces the goodness of fit of the model for all those students who used this method.

Interestingly, it seemed to be the better students who used Lami's theorem- a point which will be investigated separately later.

Identification of New Malrules

Several scripts appeared to contain a new malrule not observed in the training set. This had the form:

If a body is resting on a horizontal surface

and

the body is in limiting equilibrium

THEN

F = mu * m

This may be distinguished from Malrule 1 because subject S2, in solving Q4, gives the weight of the hanging body as "a * g", while the friction force on the stationary body is given as "mu * b" instead of "mu * b * g". If the subject had used Malrule1 (Weight = Mass), he would have given the weight of the hanging body as "a" and the friction force on the stationary body as "mu * b".

This new malrule was observed in five solutions (S2Q4V1, S2Q4V2, S6Q4V1, S6Q4V2 and S21Q4V2). From here on, it will be referred to as the "F = mu * m" malrule.

Description of the Data

With 23 subjects each presented with eight questions, there are 184 potential data points to consider. These can be broken down as follows:

Correct Solutions:	90
Incorrect Solutions:	41
Not Attempted:	53
	184

Of the solutions attempted, nine involved Lami's Theorem, and four used the angle of friction method. As these were not counted in the rest of the analysis, that left 118 solutions to be considered further.

Occurrence of Malrules

The frequency of occurrence of identified malrules in the validation set is given below. The newly identified "F = mu * m" malrule is not numbered because it was not induced from the initial training set. In the figures below, the occurrences of a malrule on a repeated question are treated separately.

	Malrule	Frequency of Occurrence
24	Unknown Force = Known Force * cos a	11
17	cos a -> sin a	7
1	Weight = mass	6
-	F = mu * m	5
4	Reaction = Weight	1
12	cos a -> a	1
	Others	0
	Total	31

The fact that 23 new subjects produced only 1 identifiable new malrule, while previously identified malrules accounted for 26/31 = 84% of the occurrences, suggests that the majority of consistent malrules in this domain have been discovered and identified. The suggestion cannot be statistically formalised because there is no reason to suppose that the probability of the next subject examined exhibiting a new malrule is independent of the previous subjects. For one thing, subjects are drawn from particular teaching groups, which vary considerably in regard to the distribution of malrules exhibited by their members.

Of the twenty-four malrules identified from the training set, nineteen were not observed in the validation set. Subjects in the training set were not asked to do the questions twice, and hence there exists no way of telling whether these errors would have been repeated consistently. It is possible that many of these errors would not have been consistent. To identify every error as a malrule would appear to claim for it a more permanent and structured mental representation than may be justified: some of the malrules identified from the training

set may in fact result from the construction of temporary ad-hoc patches similiar to those observed by Scanlon, Hawkridge and O'Shea (1983).

Hierarchies of Malrule Consistency

The difference in consistency of behaviour exhibited by problem-solvers may be used to categorise malrules into three different levels of consistency:

1 Errors observed but not repeated

2 Errors repeated in the same question but not in other questions

3 Errors repeated in other questions.

Errors of the first category would certainly appear, in the absence of other evidence, to be transient results not corresponding to permanent knowledge structures. Errors of the third category are prima facie candidates for forming part of the problem-solver's permanent skill repertoire, and their identification in a teaching situation is hence an important

objective. These errors can be definitely identified as malrules.

Errors of the second category of consistency are in some ways the most interesting group. Their repetition implies that they are remembered for at least half an hour, and that the problem-solver has some degree of confidence in them...yet they are not applied in all possible cases. In fact, in the data presented here, malrules which do not appear on every possible occasion do not appear in more than one question.

There would seem to be three possible explanations for the existence of this behaviour:

1 The second category errors have been incompletely specified- they correspond to permanent knowledge structures that operate in more specific contexts than those as yet identified.

2 They do not correspond to permanent procedural knowledge structures at all, but are temporary data structures which have been remembered, cued by the context of the identical question to that in which they were generated.

3 The problem-solver maintains several permanent problem solving procedures for dealing with the relevant principle and context. If one procedure is used on a question, it is more likely to be used again on the same question, but it is not certain to be used on other problems.

The data available do not seem sufficient to pick out one explanation unambiguously as the correct one; however, the data do relate to the relative plausibility of different explanations.

1 The idea that these are incompletely specified malrules seems a possible explanation. If so, however, the extra restrictions on the application of these malrules would need to be sensitive to the context of application in some essentially irrelevant way. Such a malrule would not closely resemble a

correct procedure, but would look more like a hybrid between a malformed correct procedure and a problem described in terms of its type hierarchy.

2 The idea that such errors are due to purely temporary data structures seems initially the most plausible, but is not unambiguously supported by the data. As will be seen later, there is a correlation between initial solutions which cannot be duplicated by the identified malrules, and solutions which are not repeated.

Errors of the second consistency category, where the error is repeated in the same question but not in other questions, differ from these solutions in being both representable as malrules, and consistent over identical questions. If they were simply ad-hoc and temporary procedure patches, there would be no reason to find a correlation between these two properties.

3 There seems to be no evidence against the explanation in terms of multiple problem-solving procedures- but to find evidence in its favour, it would be be necessary to give every subject many more than eight problems. If the malrule present in one

solution was evinced in another, this would be supporting evidence for the existence of multiple knowledge structures in the problem-solver.

Relation of Questions to Potential Malrules

When attempting to fit a model to a problem-solver's output, it is necessary to know what malrules a solution could potentially involve. This is because if a problem-solver exhibits malrule5 (say) on one question but not on another where it could be applied, the convention for model fitting described in the definition of the mu-quotient demands that the malrule should not be included in the student model, and that the solution which exhibits it be deemed a "miss".

The relationship between questions and potential malrules is as follows:-

	Q1	Q2	Q3	Q4
Malrule1	yes	yes	yes	yes
Malrule4	yes	no	yes	no
Malrule5	yes	no	no	no
Malrule12	yes	yes	yes	yes
Malrule17	no	yes	yes	yes
Malrule24	yes	yes	no	yes
F = mu * m	no	no	no	yes

In order to distinguish malrules used only on a single question from malrules used consistently over several questions, it is necessary that at least two questions should potentially exhibit each malrule. This is not the case with the newly identified malrule, and consequently it is impossible to categorise its degree of consistency for particular problem-solvers. Since it was not anticipated when the validation set was designed, it was not possible to structure the questions in such a way as to obtain this data. Malrule5 also appears in potentially just one question. In the data, this malrule is not exhibited except in a single instance of a single question, and so ambiguity of its scope is not in question.

Summary of Results

In the summary of results, solutions will be described in the following notation:

Y	correct solution
X	incorrect solution
-	no solution
[L]	done by Lami's Theorem
[a]	done by the angle of friction method
[1,2,3]	solution exhibits malrules 1,2 and 3
[-]	solution cannot be modelled
[M]	solution exhibits the malrule "F = mu * m"

Every question will be noted twice- once for each of the times it could have been attempted. Remembered answers are counted as repeated working, expressions of general formulae are ignored, and solutions are described as far as they exist (i.e- no account is taken of halting behaviour).

	Q1		Q2		Q3		Q4	
S1	Y	Y	X[-]	[-]	Y	Y	-	-
S2	Y	Y	X[24]	X[24]	X[17]	X[17]	X[M]	X[M]
S3	X[-]	-	Y	-	-	X[1]	-	-
S4	X[a]	X[a,5]	X[24]	X[1]	Y	Y[a]	-	-
S5	X[-]	X [-]	X[1,12]	-	-	-	X[-]	X[-]
S6	Y	Y	Y	Y	Y	X[a]	X[M]	X[M]
S7	X[17]	X [17]	-	-	-	-	-	-
S8	Y	Y	X[24]	X[24]	Y	Y	-	Y[L]
S9	Y	Y	-	-	Y	Y	Y[L]	Y[L]
S10	Y	Y	Y	Y	Y	Y	X[-]	X[-]
S11	Y	Y	X[24]	X[24]	-	-	-	-
S12	Y	Y	-	-	Y	Y	Y[L]	Y[L]
S13	X[1]	X [1]	Y	X [24]	Y	Y	-	-
S14	Y	X [4]	X[1]	X [24,17]	X[17]	X[17]	X[-]	X[-]
S15	Y	Y	Y	-	Y	Y	-	-
S16	Y	Y	Y	-	Y	Y	-	-
S17	Y	Y	-	-	Y	Y	-	-
S18	Y	Y	Y	Y	Y	Y	-	-
S19	Y	Y	-	-	Y	Y	-	-
S20	Y	Y	Y	Y	Y	Y	X[-]	-
S21	Y	Y	Y	Y	Y	Y	X[-]	X[M]
S22	Y	Y	Y[L]	Y[L]	Y	Y	Y[L]	Y[L]
S23	Y	Y	-	-	-	Y	-	-

This data will now be analysed in six different ways. The two measures used will be the mu-qotient, where applicable, and the error fit measure defined in Chapter V. Because the students in the validation set never used a malrule consistently on two occasions where it could be used without using it consistently on all possible equations, there are no "false errors" in any of the data analyses, and consequently the error fit measure is numerically equal to:

$$\frac{\text{hits}}{\text{data points}}$$

Analyses Used

The six analyses performed on the raw data may be summarised as in Chapter V:

Theory	Span of Consistency	Span of Comparison	Prediction
Competence	single subject	all questions correct first time	consistent correct behaviour
Consistent Errors	single subject	all consistent erroneous answers plus expected errors	consistent erroneous behaviour
Consistent Errors	single data item	as above	as above
Inconsistent Errors	-	all answers NEWT can't model	inconsistent or null behaviour
Student Model	single subject	all questions attempted first time	consistent behaviour
Student Model	single question	all questions attempted first time	consistent behaviour

In more detail, the six analyses are:

1 An analysis of competent problem-solving. For this analysis, questions solved via Lami's Theorem or the Angle of Friction method were excluded.

The span of comparison used was the set of remaining questions correctly answered at the first attempt. Answers NEWT was unable to model at the equation generation level, or which were not solved correctly the second time, were counted as misses, and answers which NEWT could model on both attempts were counted as hits.

This tests NEWT as a model of competent problem-solving. Since there are neither malrules nor false errors in this analysis, both measures are numerically equal. The span of consistency is the whole data set, since the model of correct problem-solving (equivalent to the MECHO program), is invariant throughout this analysis.

2 An analysis of consistent errors. The span of comparison is the set of questions solved erroneously in which the solution is identical at both attempts.

The span of consistency is a single subject. Hence a model is generated for each subject, giving the malrules they are identified as possessing. The newly identified malrule is not used in this analysis.

This tests the explanatory power of NEWT at dealing with consistently erroneous behaviour across different questions.

3 An analysis of consistent errors. The span of comparison is again the set of questions solved erroneously in which the solution is identical at both attempts. The span of consistency is relaxed to cover both attempts at a single question.

This tests the explanatory power of NEWT at dealing with consistent behaviour across a single question. Malrules in the second category of consistency, namely, errors which are consistent only within a single question, are counted as hits on this analysis, but not the previous one. If there were significant differences between this analysis and the previous one (and the measures could only be higher for this analysis, as all hits on the previous analysis would count as hits on this one as well); it

would be evidence that problem-solvers were more consistent when dealing with identical questions than when dealing with different questions.

4 An analysis of inconsistent errors. Here, the span of comparison is the set of solutions which NEWT is unable to model at the first attempt. The prediction based on this is that the second attempt at each question will be different from the first (and possibly non existent).

This tests the ability of NEWT to identify questions which the problem-solver will solve inconsistently. Since no malrules are introduced into NEWT for this analysis, the span of consistency is the whole data set (as for the analysis of competence), and the mu-quotient is therefore equal to the error fit measure. It is not possible to predict in advance which solutions will fall into this category, only to identify them when they occur.

5 An analysis of NEWT as a student model (single student span of consistency). Here, the span of comparison is the set of all questions attempted in the first instance.

The model predicts that if the first instance of a question cannot be modelled, the second will be inconsistent, or will not be attempted.

6 An analysis of NEWT as a student model (single question span of consistency). Here, the span of comparison is the set of all questions attempted in the first instance.

Analysis 5 identifies the category 3 malrules held by the student, and analysis 6 identifies the category 2 and category 3 malrules. For a population of wholly consistent students, these analyses would yield the same results. Recall that Category 2 malrules are errors consistent only on repeated solutions to the same question, while Category 3 malrules are consistent under all conditions.

Additionally, an analysis will be conducted on students using Lami's Theorem. These have been excluded from other analyses because NEWT does not include a procedure implementing this method, but the data on such subjects are interesting in their own right, independently of NEWT. In the following tables, "H" is used to stand for a hit (a student result corresponding to a NEWT result), and "M" is used to stand for a miss.

1. Analysis of Competent Problem-Solving

	Q1	Q2	Q3	Q4
S1	H	-	H	-
S2	H	-	-	-
S3	-	-	-	-
S4	-	-	M	-
S5	-	-	-	-
S6	H	H	M	-
S7	-	-	-	-
S8	H	-	H	-
S9	H	-	H	-
S10	H	H	H	-
S11	H	-	-	-
S12	H	-	H	-
S13	-	M	H	-
S14	M	-	-	-
S15	H	M	H	-
S16	H	-	H	-
S17	H	-	H	-
S18	H	H	H	-
S19	H	-	H	-
S20	H	H	H	-
S21	H	H	H	-
S22	H	-	H	-
S23	H	-	-	-

mu-quotient = 36/41 = 0.88

error fit measure = 36/41 = 0.88

Notice that there are no entries for Q4. This is definitely the hardest question for most subjects, and was either not answered at all (12 subjects), or answered incorrectly (7 subjects), or answered correctly by means of Lami's Theorem (4 subjects).

2. Analysis of Consistent Errors- Student Consistency Span

	Q1	Q2	Q3	Q4	Model
S1	-	-	-	-	-
S2	-	H[24]	M	M	[24]
S3	-	-	-	-	-
S4	-	-	-	-	-
S5	-	-	-	-	-
S6	-	-	-	M	-
S7	H[17]	-	-	-	[17]
S8	-	H[24]	-	-	[24]
S9	-	-	-	-	-
S10	-	-	-	M	-
S11	-	H[24]	-	-	[24]
S12	-	-	-	-	-
S13	M	-	-	-	-
S14	-	-	H[17]	-	[17]
S15	-	-	-	-	-
S16	-	-	-	-	-
S17	-	-	-	-	-
S18	-	-	-	-	-
S19	-	-	-	-	-
S20	-	-	-	-	-
S21	-	-	-	-	-
S22	-	-	-	-	-
S23	-	-	-	-	-

mu-quotient = 3/10 = 0.3

error fit measure = 5/10 = 0.5

Notice that in this analysis, only two different malrules are identified as applying over a whole span of consistency (one student). These are Malrules 17 and 24. All the other malrule instances are not counted as part of the model because they would lead to false errors.

Thus, for example, S2 is not counted as having malrule17 in their student model, because malrule17 could be exhibited in Q2 and Q4. Hence its inclusion would lead to one more hit, but two false errors.

3. Analysis of Consistent Errors- Question Consistency Span

	Q1	Q2	Q3	Q4
S1	-	-	-	-
S2	-	H[24]	H[17]	M
S3	-	-	-	-
S4	-	-	-	-
S5	-	-	-	-
S6	-	-	-	M
S7	H[17]	-	-	-
S8	-	H[24]	-	-
S9	-	-	-	-
S10	-	-	-	M
S11	-	H[24]	-	-
S12	-	-	-	-
S13	H[1]	-	-	-
S14	-	-	H[17]	-
S15	-	-	-	-
S16	-	-	-	-
S17	-	-	-	-
S18	-	-	-	-
S19	-	-	-	-
S20	-	-	-	-
S21	-	-	-	-
S22	-	-	-	-
S23	-	-	-	-

mu-quotient = 4/10 = 0.4

error fit measure = 7/10 = 0.7

At this level of consistency, there are of course more hits, and the number of malrules identified has risen from two to three.

4. Analysis of Inconsistent Errors

	Q1	Q2	Q3	Q4
S1	-	H	-	-
S2	-	-	-	M
S3	H	-	-	-
S4	-	-	-	-
S5	H	-	-	H
S6	-	-	-	M
S7	-	-	-	-
S8	-	-	-	-
S9	-	-	-	-
S10	-	-	-	M
S11	-	-	-	-
S12	-	-	-	-
S13	-	-	-	-
S14	-	-	-	H
S15	-	-	-	-
S16	-	-	-	-
S17	-	-	-	-
S18	-	-	-	-
S19	-	-	-	-
S20	-	-	-	H
S21	-	-	-	H
S22	-	-	-	-
S23	-	-	-	-

Since no student models were constructed for this analysis, and therefore no malrules were incorporated into student models,

mu-quotient = 7/10 = 0.7

The error fit measure is not applicable to this analysis, because only questions incorrectly answered are included in the data. This makes the concept of a 'false error' inappropriate.

An alternative approach to calculating the mu-quotient is to look at the comparison between error questions solved inconsistently (x), and solutions NEWT cannot model (y).

For this comparison, x takes a value of 1 for every question solved inconsistently, and zero for every question solved consistently. The value of y is taken as 1 for every erroneous solution NEWT can't model in the first instance, and zero otherwise. This gives the following distribution of (x,y) values:

	Q1	Q2	Q3	Q4
S1	0,0	1,1	0,0	-
S2	0,0	0,0	0,0	0,1
S3	1,1	1,0	-	-
S4	-	1,0	0,0	-
S5	1,1	1,0	-	1,1
S6	0,0	0,0	1,0	0,1
S7	0,0	-	-	-
S8	0,0	0,0	0,0	-
S9	0,0	-	0,0	-
S10	0,0	0,0	0,0	0,1
S11	0,0	0,0	-	-
S12	0,0	-	0,0	-
S13	0,0	1,0	0,0	-
S14	1,0	1,0	0,0	1,1
S15	0,0	1,0	0,0	-
S16	0,0	1,0	0,0	-
S17	0,0	-	0,0	-
S18	0,0	0,0	0,0	-
S19	0,0	-	0,0	-
S20	0,0	0,0	0,0	1,1
S21	0,0	0,0	0,0	1,1
S22	0,0	-	0,0	-
S23	0,0	-	-	-

These results may be summarised in this form:

		y (NEWT) 1	0	
x	1	7	9	16
(student)	0	3	44	47
		10	53	63

If we assume that the probability of a student scoring a 1 or a zero is independent of the probability that NEWT does the same, we can calculate the expected frequencies given the row and column totals as:

		y 1	0	
x	1	2.54	13.46	16
	0	7.46	39.54	47
		10	53	63

Using Fisher's Exact Test, the probability of obtaining a distribution at least as different from the expected one as this is 0.0031. Thus it is highly likely that the scores of NEWT and the student population are related- that is, if NEWT cannot model a student's output, then the student is likely to be inconsistent; while if NEWT can model their output, they are likely to be consistent themselves.

5. Analysis of NEWT as a Student Model

(Single Student Consistency Span)

For this analysis, the span of consistency is one student, and the span of comparison is the first instance of each question. When the first instance cannot be modelled by NEWT, the prediction made by the student model is that the second instance of the question will not be attempted. Predicted correct solutions are counted as hits.

	Q1	Q2	Q3	Q4	Model
S1	H	H	H	-	-
S2	H	H	M	M	[24]
S3	H	M	M	-	-
S4	-	M	-	-	-
S5	M	M	-	M	-
S6	H	H	-	M	-
S7	H	-	-	-	[17]
S8	H	H	H	-	[24]
S9	H	-	H	-	-
S10	H	H	H	M	-
S11	H	H	-	-	[24]
S12	H	-	H	-	-
S13	M	M	H	-	-
S14	M	M	H	M	[17]
S15	H	M	H	-	-
S16	H	-	H	-	-
S17	H	-	H	-	-
S18	H	H	H	-	-
S19	H	-	H	-	-
S20	H	H	H	H	-
S21	H	H	H	M	-
S22	H	-	H	-	-
S23	H	-	M	-	-

mu-quotient = 42/62 = 0.68

error fit measure = 44/62 = 0.71

6. Analysis of NEWT as a Student Model

(Single Question Consistency Span)

For this analysis, the span of consistency is one question, and the analysis is otherwise identical to that carried out above.

	Q1	Q2	Q3	Q4	Model
S1	H	H	H	-	-
S2	H	H	H	M	[17,24]
S3	H	M	M	-	-
S4	-	M	-	-	-
S5	M	M	-	M	-
S6	H	H	-	M	-
S7	H	-	-	-	-
S8	H	H	H	-	[24]
S9	H	-	H	-	-
S10	H	H	H	M	-
S11	H	H	-	-	[24]
S12	H	-	H	-	-
S13	H	M	H	-	[1]
S14	M	M	H	M	[17]
S15	H	M	H	-	-
S16	H	-	H	-	-
S17	H	-	H	-	-
S18	H	H	H	-	-
S19	H	-	H	-	-
S20	H	H	H	H	-
S21	H	H	H	M	-
S22	H	-	H	-	-
S23	H	-	M	-	-

mu-quotient = 43/62 = 0.69

error fit measure = 46/62 = 0.74

Analysis of the Use of Lami's Theorem

Both groups of subjects had been taught Lami's Theorem. Subjects S8, S9 and S12 in the first group used it, and S22 in the second group. Since NEWT had not been given a procedure for using this method, solutions involving it have not been considered in previous analyses.

It is easy to see that the students who used this method were better at problem-solving in general than the rest of the subjects. The people who used Lami's Theorem attempted 27 questions and got 25 correct (ie 93%). The people who did not use Lami's Theorem attempted 104 questions and got 64 correct (ie 62%). Even more strikingly, 100% of the questions done by Lami's theorem were done correctly, as against 66% of the questions done by other methods. This is even though 77% of the questions done by Lami's Theorem were question 4, the most difficult question of all

In order to test the hypothesis that subjects using Lami's theorem were better problem-solvers than subjects who did not, a non-parametric test dealing with unequal sample sizes is required. The Mann-Whitney U test is therefore an appropriate test to use- although the number of people using Lami's theorem are distinctly on the low side for obtaining significant results.

Dividing the population into subjects who do use Lami's Theorem, and subjects who don't, the two relevant hypotheses are:

H0: the two populations have identical distributions.

H1: The population using Lami's Theorem has a higher median score.

The data for the calculation of this statistic are as follows:

Subject	Score	Use of Lami	Rank
S1	4		12.5
S2	2		17.5
S3	1		20
S4	1		20
S5	0		22.5
S6	6		5
S7	0		22.5
S8	5	Y	9.5
S9	6	Y	5
S10	6		5
S11	2		17.5
S12	6	Y	5
S13	3		15.5
S14	1		20
S15	5		9.5
S16	4		12.5
S17	4		12.5
S18	6		5
S19	4		12.5
S20	6		5
S21	6		5
S22	8	Y	1
S23	3		15.5

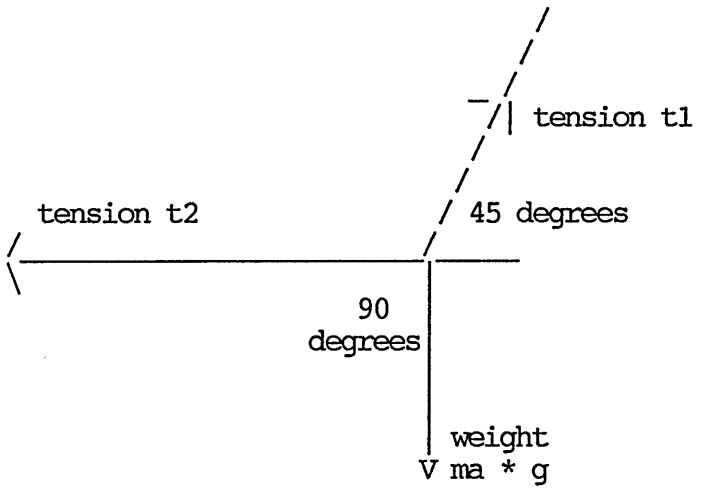
For this data: U1 = 10.5 and U2 = 65.5

The smaller of these is taken as the test statistic: U = 10.5

Since the 0.025 significance level for n1 = 19 and n2 = 4 is 13, the alternative hypothesis is rejected, and the population of subjects using Lami's Theorem scores significantly higher than the population of subjects not using it.

These figures support a correlation between expertise and the use of Lami's Theorem: either experts are more likely to use it than novices, or else students are only able to learn this method after developing a reasonable degree of expertise already.

Such a correlation can be looked at in the light of ideas on planstacking developed earlier. The advantage of Lami's Theorem is that it allows what would otherwise be two stages of working to be condensed into one. Consider this physical situation:



Without use of Lami's Theorem, one can find t2 in terms of ma and g by resolving vertically and horizontally:

t1 * sin 45 = ma * g

t2 = t1 * cos 45 = (ma * g * cos 45) / sin 45

t2 = ma * g

Whereas using Lami's Theorem, these two stages can be collapsed into one:

$$\frac{t2}{\sin(90 + 45)} = \frac{ma * g}{\sin(90 + 45)}$$

$$t2 = ma * g$$

Anyone who was consciously planning a solution, and hoping to minimise the number of steps used, would have a motive for selecting Lami's Theorem as the method of choice. Anyone who was solving the problem by applying whatever methods could be applied until an answer emerged would have no more reason to use this method than any other.

This finding therefore supports the idea that experts plan their solutions qualitatively before implementing them- which is the essence of the planstacking approach.

Summary of Analyses

The results of the first six data analyses may be summarised thus:

Analysis	Span of Comparison	Span of Consistency	Mu-quotient	Error Fit Measure
Competent Problem-Solving	Questions correct first time	whole data set	0.88	0.88
Consistent Errors	consistent incorrect questions	one student	0.3	0.5
Consistent Errors	consistent incorrect questions	one question	0.4	0.7
Inconsistent Errors	questions which can't be modelled	whole data set	0.70	N/A
Student Model	All questions	one student	0.68	0.71
Student Model	All questions	one question	0.69	0.74

Effects of Considering New Malrules

If the new malrule identified from the validation set had been implemented in NEWT, and included in the analysis, the figures would have been slightly different. Revised figures taking the new malrule into consideration are as follows:

Analysis	Span of Comparison	Span of Consistency	Mu-quotient	Error Fit Measure
Competent Problem-Solving	Correct questions	whole data set	0.88	0.88
Consistent Errors	Incorrect consistent questions	one student	0.4	0.7
Consistent Errors	Incorrect consistent questions	one question	0.5	0.9
Inconsistent Errors	Questions which can't be modelled	whole data set	0.87	N/A
Student Model	All questions	one student	0.7	0.73
Student Model	All questions	one question	0.71	0.76

As can be seen, the values change only marginally when the new malrule is included in the analysis.

Alternative Predictions of Behaviour for the Student Model

Since the analysis of NEWT as a student model involves the prediction of behaviour, it may be compared to the alternative hypothesis that the subject will repeat the solution to every question on the second attempt. This is here termed the 'Repeat Hypothesis'.

Analysis of the Repeat Student Model

	Q1	Q2	Q3	Q4
S1	H	M	H	-
S2	H	H	H	H
S3	M	M	M	-
S4	-	M	-	M
S5	M	M	-	M
S6	H	H	-	H
S7	H	-	-	-
S8	H	H	H	-
S9	H	-	H	-
S10	H	H	H	H
S11	H	H	-	-
S12	H	-	H	-
S13	H	M	H	-
S14	M	M	H	M
S15	H	M	H	-
S16	H	-	H	-
S17	H	-	H	-
S18	H	H	H	-
S19	H	-	H	-
S20	H	H	H	M
S21	H	H	H	M
S22	H	-	H	-
S23	H	-	M	-

mu-quotient - not applicable

Error fit measure: 47 / 62 = 0.76

The different possible student models available for prediction of behaviour may be summarised thus:

Analysis	Malrules	Span of Consistency	Mu-quotient	Error Fit Measure
NEWT student model	Original set	Student	0.68	0.71
NEWT student model	Original set	Question	0.69	0.74
NEWT student model	Including new rule	Student	0.7	0.73
NEWT student model	Including new rule	Question	0.71	0.76
Repeat model	-	Question	-	0.76

This table shows that NEWT used as a student model is comparable in predictive power to the Repeat Hypothesis. On a question by question basis, NEWT augmented by the new malrule is as good as the Repeat Hypothesis- yet it can yield information not available from the alternative; namely, predictions about solutions for previously unseen questions. These predictions (student level of consistency), are not so accurate because students themselves are less consistent between questions than when repeating the same question.

Unlike the previous data analyses, the mu-quotient and the Error Fit measure for the student model may be contrasted with a Chi-squared analysis. For this analysis, the Repeat Model is taken as the alternative hypothesis, and the number of hits by that model is taken as the observed frequency. The student model version of NEWT is taken as the null hypothesis, and the number of hits by that model is taken as the expected frequency. Since there is a single class with no constraints, the number of degrees of freedom is one.

For NEWT as a student model using a single student span of consistency,

Chi-squared = 0.205

For NEWT as a student model, using a single question span of consistency,

Chi-squared = 0.022

Since both of these values are less than 3.84, the 5% significance level, either version of the NEWT Student Model may be considered to be comparable to the Repeat Hypothesis as a predictor of behaviour.

The real advantage of a NEWT-based student model over the Repeat Model is of course that it enables predictions to be made of student performance on problems not previously encountered.

Conclusions

This chapter has considered the experimental data as presented in the appendix to this chapter, and has presented a number of statistical analyses on them. The results of these analyses show that:

- Students who use Lami's Theorem are better at solving problems than students who don't.

- NEWT used as a student model is comparable in predictive ability to the Repeat Hypothesis (ie, the hypothesis that students repeat their solutions to identical questions)

- The new malrule identified during the analysis of the training set does not affect the figures very much.

- Most of the consistent errors were due to a handful of malrules, and most of the malrules incorporated in NEWT were not needed in the analysis of the validation set.

- There was a significant correlation between incorrect solutions that NEWT could not model, and solutions that were handled inconsistently between the two instances of the question. This shows that errors that do not correspond to NEWT malrules are not caused by permanent knowledge structures held by the problem-solver.

- There was a broad agreement between the results analysed by the Error Fit measure and the Micro-Theory Evaluation Quotient over the analysis for which they were used.

- The fact that one new malrule was discovered in 23 scripts from two separate teaching groups suggests that collection of data from more groups might lead to the identification of more malrules- but not very many, and not sufficient to affect the figures very much. It is not possible to formalise this prediction, because there is no reason to suppose that the distribution of malrules is independent of teaching groups.

The next chapter will consider the relevance of these results to the theories discussed earlier.

Chapter V111 - Conclusions

This project has identified three psychological mechanisms used by human problem-solvers, and has resulted in the construction of a program whose output corresponds reasonably well with consistent human problem solutions.

Aims of the Project

The project has two aims:

(1) To identify the psychological mechanisms used by problem-solvers working in the domain.

(2) To develop a model that allows a student's future behaviour to be predicted from their past actions.

The first of these aims has been pursued by an analysis of published work, and the second by the construction of a problem-solving program called NEWT. This has been compared with the problem solutions of twenty-three students, each attempting to solve four problems twice.

Claims of the Project

Three mechanisms for solving problems have been identified:

- a control strategy termed "Planstacking", used by expert problem-solvers.

- a heuristic termed the "Hidden Curriculum Assumption of Non-Redundancy", used by expert problem-solvers to reduce the amount of search needed.

- an intermediate knowledge representation termed a "sketch", used by experts and novices alike to disambiguate syntactic parses of questions.

A computational model has been constructed and analysed. The conclusions drawn from this are:

- the NEWT program is an effective and accurate basis for a student model.

- the NEWT program is reasonably successful at modelling consistent student errors. This demonstrates the existence of coherent and stable erroneous procedures used by students, termed "malrules". Six malrules are identified.

- questions which are answered inconsistently largely correspond to questions NEWT was unable to model. This is taken as showing that some errors are not due to coherent and stable procedures, but are ad-hoc results produced by random manipulations of quantities in the problem. This correlation allows NEWT to identify problems whose solution has been obtained by a weak general process, as well as identifying malrules when they occur.

Identification of Psychological Mechanisms

Planstacking

How do problem-solvers know what principle to apply next in order to generate an equation? Forward inference and backward inference have been suggested as possible control strategies. In this thesis, a third alternative called "Planstacking", is proposed. This involves the construction of successive plans which are placed on a stack as they are created. Each plan consists of a physical principle and a context in which it can be applied. Each plan is capable of generating an equation, and if the resultant equation contains only one unknown quantity, the plan therefore defines a means of finding that quantity. Plans are generated to find sought quantities by backward inference. When plans to solve for all sought quantities have been generated, the stack is popped and the equations are generated and solved in reverse order.

It is hypothesized that this method can only be applied by people able to use meta-level inference to tell what quantity could be inferred from a plan. Therefore, this method is only available to experts.

The proposal explains existing experimental data on problem-solving better than alternative hypotheses, though as yet only in a qualitative way.

The Hidden Curriculum Assumption of Non-Redundancy

Evidence suggests that expert problem-solvers can often guess the method of solution for problems they have not read completely. To explain this, the Hidden Curriculum Assumption of Non-Redundancy is introduced. This is an assumption made by expert problem-solvers to the effect that every quantity mentioned in the problem is relevant. Such an assumption allows experts to rank their list of known physical principles in order of their likely utility in solving the problem. Principles that relate all the quantities mentioned in the problem statement so far are given priority over others.

This ordering reduces the amount of search needed later on in the solution process for problems that contain no redundant information. It also gives a basis for an expert to guess the relevant solution principle before reading the whole question.

The Use of Sketch Construction to Parse Questions

When given a question to solve in which the objects described cannot physically exist, some subjects re-parse the question before solving any equations. These subjects construct a representation of a possible object which does not correspond to the description, rather than an impossible object which does correspond to the description.

This is explained by the hypothesis that subjects construct either internal or external sketches of problem contexts as they are described. A sketch is an arbitrary metric instantiation of the problem description, and its construction verifies the possibility of the objects described in it. Failure to construct a sketch is taken as indicating an incorrect parse of the incoming description; leading to abandonment of the problem, or forcing re-parsing of the question.

The NEWT Model of Problem-Solving

The NEWT program was constructed to model erroneous human problem-solving at the level of equation generation. The MECHO program (Bundy et al. 1979), which was able to solve problems correctly, was adapted to produce errors corresponding to the malrules defined in Chapter VI.

NEWT was compared with twenty-three students solving four problems twice. Sometimes students solved the problems consistently (produced the same results both times), and sometimes they solved the problems inconsistently (and produced different results on the two occasions).

NEWT was compared to the students over:

(a) the whole set of questions answered, to evaluate its merit as a student model,

(b) the set of questions answered consistently, to evaluate it as a model of consistent problem solution,

and (c) the set of questions answered inconsistently, to investigate the relationship between inconsistent problem solutions and the NEWT model.

For each analysis, the "span of consistency", is the set of data points over which the NEWT model is held constant.

Summary of Results

	Analysis	Span of Consistency	Mu-quotient	Error Fit Measure	Other Measures
(a)	Student Model	Single Student	0.68	0.71	not significantly different from Repeat Hypothesis
(b)	Consistent Errors	Single Student	0.3	0.5	-
(c)	Inconsistent Errors	Whole data set	0.7	-	probability of chance result = 0.0031

These figures show that NEWT was able to duplicate about 70% of the students' behaviour overall; about 50% of their consistently erroneous behaviour, and to recognise (but not duplicate) about 70% of their inconsistent behaviour.

This provides an adequate basis for a student model in a Computer-Assisted tutorial program, and provides additional evidence to support the existence both of a limited number of principled and consistent errors, or "malrules", and of weak general methods that lead to inconsistent errors on the part of the student.

Malrules Identified

The following malrules were identified in the training set, and confirmed in the validation set:

	Malrule	Frequency of Occurence
24	Unknown Force = Known Force * cos a	11
17	cos a --> sin a	7
1	Weight = Mass	6
4	Reaction = Weight	1
12	cos a --> a	1

In addition, one new malrule was inferred from the validation set:

25	Friction = mu * Mass (mu is the coefficient of friction)	5

While nineteen other errors were identified in the training set, the figures suggest that these six are sufficient to account for most of the consistent errors likely to be encountered. It is interesting that these errors have the appearence of malformed formal procedures, rather than processes derived from naive physical intuitions. This suggests that the mental activities of naive and formal physics are largely disjoint, at least in the domain investigated here.

Limitations of the Project

The work carried out in this project has a number of limitations. These are:

Computational Implementation

Experimental Validation

Methodological Incompleteness

Extension of Data Collection

Interpretation of Behaviour

The Assignment of Credit Problem

Causation of Inconsistency

Utility of the Student Model

Computational Implementation

The hypotheses advanced regarding Planstacking, the Hidden Curriculum Assumption of Non-Redundancy, and Sketch Construction; have been articulated discursively, but have not been given a computational implementation.

Until and unless this is done, it will be impossible to be certain whether these defined procedures are capable of producing the expected results without non-trivial extensions in their structure. The discipline of computational implementation is a demanding one, but it is ultimately vital for the validation of hypotheses of this type.

Experimental Validation

The hypotheses on problem representation and control strategy described above have been related to published results of other researchers, but have not been tested here by comparison with experimental data collected for the purpose. Several predictions of behaviour can be derived from these hypotheses, which should therefore be expected to gain observational corroboration before they can be regarded as well-established. An attempt to compare the relative merits of Planstacking and Schema-Guided Forward Inference has been carried out by Priest and Lindsay (1986).

These authors carried out two studies to compare the predictions of the different control strategies with the behaviour of experimental subjects. In the first study, they investigated the latency of problem solution over different versions of the same problem; and in the second study, they looked at the extent to which different subjects were able to produce explicit plans to solve problems. These will be referred to here as Study A and Study B.

Study A: Latency on Different Problems.

Consider a pair of problems, referred to as the "Study Version", and the "Test Version". Suppose that the subject is required to study the Study Version, and then to solve the Test Version. In such a situation, performance on the Test Version should be related to the amount of processing carried over from the Study Version. We shall assume that the two questions are identical as to the objects described in them, but that the Test Version may or may not vary from the Study Version in terms of the quantities sought or given. If the Test Version is identical to the Study Version, we shall refer to it as the "Standard Version". If the Test Version is the same as the Study Version except that the quantity sought in the problem is different, we shall refer to it as the "Changed Soughts Version"; and if only the given quantities have been changed, we shall refer to it as the "Changed Givens Version". Now consider the predictions of the different control strategies as regards the solution of a Test Version question by a subject:

> A problem-solver working by Schema-Guided Forward Inference would be able to use all their previous working on the Standard Version, some of it on the Changed Soughts question, and none of it on a Changed Givens question. Therefore, performance should be better on a Standard Version than a Changed Soughts Version; and better on a Changed Soughts Version than on a Changed Givens question.
>
> A problem-solver working by Planstacking would be able to use all their previous working on a Standard Version, some of it on a Changed Givens question, and none of it on a Changed Soughts question. So they ought to be able to solve a Standard Version better than a Changed Givens Version, and a Changed Givens Version better than a Changed Soughts Version.

Thus the two control strategies lead to different predictions in solving slightly different questions. Improved performance on a particular question may take the form of either faster, or more accurate problem solution, since the student has the option of making a tradeoff between speed and accuracy. Thus significant interaction between either speed or accuracy and problem version is predicted by each of the two control strategies under consideration.

Seventy-nine subjects with a variety of levels of expertise were given timed problem sheets to complete, containing pairs of problems. In each pair, the first problem was the Study Version; and the second was either a Standard Version, a Changed Soughts Version, or a Changed Givens Version. The experiment was planned as a balanced repeated measures design; with level of expertise and problem version as independent variables, and solution time and inference direction as dependent variables.

The results showed no effects of version of problem on latency. There was an effect of problem version on accuracy, due to increased accuracy on Standard Version questions as predicted by both control strategies. This effect was not present when Changed Givens and Changed Soughts versions were considered in isolation.

Study B: Planning Problem Solutions

It is inherent in the Planstacking approach that the problem-solver builds up a detailed plan for solving the problem before generating any equations. Problem solution by Schema-Guided Forward Inference does not require such a step. Therefore an ability on the part of experts to write down explicit plans for

solving problems without generating equations would support the existence of a Planstacking control strategy rather than a Schema-Guided Forward Inference strategy. An inability to produce such plans would have to be counted as conclusive evidence against the use of Planstacking.

Seventy-nine subjects were given a series of problems to solve, and asked to write down an explicit solution plan without generating any equations. The result was that experts were significantly more likely to be able to produce solution plans than novices.

Thus the experimental evidence is not conclusive. The planning experiment provides support for the use of a planstacking control strategy; but the latency experiment does not, although the increased accuracy of subjects on the Standard Version problems suggests that the experimental design was valid. The results of Experiment A could be due to the use of another control strategy altogether, or to the effects of an unsuspected confounding variable. Alternatively, it could be that when subjects recognise that the question they are answering differs from the Study Version, they decide not to make use of their previous working. Thus there is a limited amount of support for the Planstacking control strategy, but none at all for the use of Schema-Guided Forward Inference.

Methodological Incompleteness

The methodology for evaluating micro-theory systems has been inadequate, and recognised as inadequate, for some time. The definition of the micro-theory evaluation coefficient is an attempt to find an answer to this problem. Six desirable criteria for such a measure have been defined, five of which are satisfied by this new statistic. The two respects in which it falls short of a complete answer to what is needed are:

- there is no value of the statistic that can be taken to correspond to a "good fit" between the micro-theory system and the data. Thus the mu-quotient cannot of itself be used to answer the question "is the theory a good fit to the data?"

- the mu-quotient does not take account of whether different micro-theories are very different or not. If some micro-theories were closely similar, and others were very different, this fact would not be reflected in the calculation of the mu-quotient.

Conventional statistical techniques often give the superficial impression of giving absolute measurements as to whether a theory is a "good fit", to some data. This is possible because in most conventional statistical hypothesis testing, there is an obvious "alternative hypothesis", against which the proposed hypotheses can be measured. In micro-theory system evaluation, there is no such alternative hypothesis .

The malrules identified in the two experimental data sets are organised into five groups, according to the correct procedure they replace. Malrules of the same group may be considered to be closer together than malrules in different groups - but this structure is not made use of by the mu-quotient.

Extension of Data Collection

Data was collected in two sets: a training set of 28 subjects, and a validation set of 23. Although these numbers seemed more than adequate in advance, they proved to be a limitation in three ways:

- the questions in the training set were not repeated. This led to the inclusion of transient errors in the set of implemented malrules.

- the analysis of consistent errors in the validation set was hampered by the small number of data points (10), compared with the number of malrules relevant to the analysis. The micro-theory evaluation quotient is not an appropriate measure to use in such cases, yielding extremely low values irrespective of the adequacy of the micro-theory system under evaluation.

- the validation set threw up one more malrule which was used with sufficient consistency to merit inclusion in the malrule set. Ideally, a new and updated version of NEWT, incorporating this new rule, should be compared

with a fresh data set for validation, large enough that an analysis of consistent errors could take place over a sufficient set of data points. This has not taken place.

Interpretation of Behaviour

Although the production of output from a problem, given a student model as a set of malrules has been automated by NEWT; the initial identification of malrules and the comparison of student output with NEWT output are done manually.

In a computer-assisted learning program, automatic identification of errors in student output in terms of a model would be indispensable. The issues involved in such identification are non-trivial, especially if an efficient process is required, but they have not been addressed in this project.

The Assignment of Credit Problem

One of the major practical problems in analysing student output for the production of a student model is the assignment of credit problem. If a student produces no output, or halts before finishing the solution, how is the "credit" for this behaviour to be assigned amongst the various processes involved? This is an interesting theoretical issue, but the project has not addressed it.

Conceivably, halting behaviour could be due to an absent or malformed control strategy, to an ignorance of relevant physical principles, to an absence of the skills of meta-level analysis that relate abstract physical principles and specific contexts with the physical quantities they relate, or to an absent procedure for forming equations from a selected principle.

If the procedural structure of human problem-solving skills is as delineated in this thesis, these are the obvious reasons for halting behaviour as opposed to the production of erroneous equations. Confirmation or further analysis would, however, require a different experimental approach from that used in this project.

Causation of Inconsistency

A separate analysis has been made of inconsistent problem-solving, and the results have been explained in terms of the existence of either semi-permanent knowledge structures corresponding to malrules, or the actions of a weak general-purpose answer-generating strategy. These two alternatives have been tentatively related to different degrees of confidence on the part of the experimental subject in the correctness of his answer, and to the tendency of solutions to be remembered or repeated.

The existence of a weak general question-answering strategy finds confirmation in other work, but there is no explicit data collected in this project that identifies such a strategy or relates it to the correlated distinctions of consistent vs inconsistent, or modellable vs unmodellable behaviour.

Any explanation for this correlation is therefore presented as a plausible hypothesis consonant with the known facts, rather than as one backed by any degree of evidential proof.

Utility of the Student Model

There are a number of limitations on the practical utility of the student model as implemented in the NEWT program.

-NEWT does not analyse student input itself.

-NEWT does not attempt to model halting behaviour.

-The problem domain covered is rather narrow. The MECHO program on which NEWT is based is capable of dealing with a wider range of problems than NEWT, but malrules have not been analysed, identified, and implemented outside the domain described here.

-NEWT only identifies incorrect procedures- it has no explicit representation for identifying which correct problem-solving procedures were used in a particular problem solution, though it would not be hard to extend NEWT in this direction, since the underlying MECHO program contains the necessary correct procedures for the domain (apart from those necessary to implement Lami's Theorem).

A student model for a CAL program would need to be able to identify the student's correct processes as well as their incorrect ones.

Further Work

Further work is suggested by the investigations recorded here. Possible lines of development could be:

1 - Computational implementation of the planstacking control strategy

2 - Computational implementation of the Hidden Curriculum Assumption of Non-Redundancy

3 - Computational implementation of a sketch construction algorithm and its interfacing with a natural language parser to investigate the use of sketches in disambiguating syntactical parses.

4 - Collection of experimental data in order to test the predictions of the Planstacking hypothesis, the Hidden Curriculum Assumption of Non-Redundancy, and the Sketch Construction Hypothesis.

5 - Extension of the coverage of NEWT to deal with erroneous problem solutions over a more widely extended domain.

6 - Repetition of the validation set data collection for the existing version of NEWT augmented by the newly identified malrule- using a larger sample size if possible.

7 - Construction and computational implementation of an algorithm for interpreting student output and identifying a malrule-based student model from it.

8 - investigation of the halting problem by protocol collection, interviews or other techniques.

9 - investigation into the psychological correlates of inconsistent problem-solving and the consistency of malrules over time by means of longitudinal studies.

10 - extension of the student model to include a structured representation of the procedures used in correct problem-solving.

11 - Investigation of the genesis of malrules by generalising the ideas of Repair Theory to appy to the domain covered by NEWT. This would involve restructing the problem-solving procedures so that each section deleted would result in the generation of set of possible malrules. The generation of all and only the malrules observed in practice would be evidence for the psychological accuracy of the structure of the initial procedure.

The domain described in this thesis is both richer and more complex than that of the subtraction algorithms for which Repair Theory was initially introduced. This may mean that a Repair Theory for this domain needs to be more complex and less knowledge-free than that adumbrated by Van Lehn. Such a Repair Mechanism should not be confused with a knowledge-free general question-answering strategy- this of its nature would not be expected to yield experimentally verifiable results.

Jansweijer, Elshout and Wielinga (1986) have applied the concept of Repair Theory to the task of solution planning in the closely related domain of Thermodynamics; but have not investigated its appliction to the generation of specific equations.

Current Activities

Of these possible lines of development, numbers 1,2,4,10 and 11 are currently being followed up by two projects being undertaken at Oxford Polytechnic. One project reported by Priest and Lindsay (1986) involves the collection of experimental data to test the predictions of the Planstacking Hypothesis, Hidden Curriculum Assumption of Non-Redundancy, and the Sketch Construction Hypothesis. Another project is based on the construction of a problem-solving program for the domain that will implement the Planstacking algorithm, Hidden Curriculum Assumption of Non-Redundancy, and also will hopefully contain a Repair mechanism enabling the generation of observed malrules from incomplete domain-specific procedures. For an approach to the development of a Repair Mechanism for the domain, see Priest (1987).

It is clear from the results of this project and the findings of other researchers that there are still more questions than answers when it comes to describing formal problem-solving. There is a need both for a rigorous and testable psychological basis for describing such behaviour, and also for a useful application of such knowledge to educational practice. This project has attempted to combine and relate these two aims.

I believe that the most fruitful approach to these questions is to try to maintain an even balance between the two approaches, and the projects currently underway at Oxford Polytechnic attempt to maintain such a balance.

In 1980, Larkin wrote:

"...there is still an enormous gap between the detailed limited theory-based tests we perform in cognitive psychology and the broad important tasks addressed by science educators. In the long run I think we shall bridge this gap with instruction that is tightly connected to theories, well tested in the laboratory, much as the design of a bridge is tightly connected to the well tested theories of mechanics. But tight connection between good theory and useful applications is probably at least five years in the future".

Now, six years later, the aims expressed are as valid as ever, and the evidence at hand strongly suggests that such a goal is ultimately possible. But tight connection between good theory and useful applications is probably still five years in the future.

Appendix I

Implementation of the Control Strategy

In order to relate the control strategy to the working program, the code used in NEWT is included here. This is presented in accordance with the methodological approach described in Chapter V. Since the theoretical content of this thesis is contained in the natural language descriptions, it is necessary to relate these to the program that implements them in order to justify the validity of the experimental results as a test of the theory.

For simplicity, trace messages have been omitted, and for ease of reference, line numbers have been added (PROLOG does not use line numbers).

```
1   geteqns ( [],Gs,Types,Us,true,[] ).

2   geteqns ( [X|Xs],Gs,Types,Us,( E&Es ),[X|Xs1] )
```

```
3           :-

4           flag( ccflag,_,off ),

5           chooseeqn( X,Types,Ez,U,Us ),

6           cleanup( Ez,E ),

7           wordsin( E,Ws ),

8           memberchk( X,Ws ),

9           union( [X|Xs],Gs,Ys ),

10          subtract( Ws,Ys,[] ),

11          !,

12          geteqns( Xs,[X|Gs],Types,[U|Us],Es,Xs1 ).

13  geteqns( [X|Xs],Gs,Types,Us,( E&Es ),[X|Xs1] )

14          :-

15          flag( ccflag,_,on ),
```

```
16          chooseeqn( X,Types,Ez,U,Us ),

17          cleanup( Ez,E ),

18          wordsin( E,Ws ),

19          memberchk( X,Ws ),

20          union( [X|Xs],Gs,Ys ),

21          subtract( Ws,Ys,Zs ),

22          append( Xs,Zs,Nxs ),

23          get_types( Zs,Types,Ntypes ),

24          geteqns( Nxs,[X|Gs],Ntypes,[U|Us],Es,Xs1 ).

25  chooseeqn( Q,Types,Eqn,Strategy,Used )

26          :-

27          finddesc( Q,Qtype,Pred,Args ),

28          applicable_formulae( Qtype,Types,Formulae-list ),
```

```
29          member( Formula,Formulae-list ),

30          plan( Formula,Q,Qtype,Pred,Args,Strategy ),

31          indep( Strategy,Used ),

32    makeeqn( Strategy,Eqn ).
```

Comments on the Code

When NEWT is asked to solve a problem, the procedure 'geteqns' is called. This has six arguments:

Argument 1: a list of quantities to be found (the 'soughts'). This will be instantiated when the procedure is called.

Argument 2: a list of quantities to be taken as given, (the 'givens'). This will be instantiated when the procedure is called.

Argument 3: a list containing the types of the soughts and givens, i.e. all the quantities mentioned in the question. This will be instantiated when the procedure is called.

Argument 4: a list of plans already used. This will be an empty list when the procedure is called.

Argument 5: a conjunction of equations generated. This will initially be empty when the procedure is called, and will contain the output of the system when the search of the problem space is completed. A conjunction acts like a list, with 'true' playing the role of the delimiting element.

Argument 6: this is an empty list when the procedure is first called. As an equation is generated to solve for a quantity, this quantity is added to the list. When the procedure is complete, the list contains all the quantities solved for by equations- either because they were originally sought or because they were introduced during the solution process.

The procedure 'geteqns' consists of three clauses (line 1, lines 2-12, lines 13-24). The first clause ensures that if the list of soughts is empty, the procedure terminates without generating any more equations.

The second clause attempts to deal with the case when there are

remaining soughts, without introducing any new unknowns. If this cannot be done, the third clause of 'geteqns' is tried. This allows the creation of new sought quantities, but is otherwise similar to the second clause.

The Second Clause:

Line 2: this is the head goal of the clause. When it is called, its inputs will be:

- a list of soughts in Argument 1. The PROLOG pattern matcher will bind 'X' to the first sought, and 'Xs' to the list of remaining soughts (which may be an empty list).

- a list of given quantities in Argument 2 (if any).

- a list of types of sought or given quantities in Argument 3.

The outputs of this clause will be:

- a list of plans used to construct equations in Argument 4.

- a conjunction of equations in Argument 5.

- a list of the quantities the equations were introduced to solve for in Argument 6.

Line 3: this is the symbol that separates the head goal of the clause from the subgoals which it calls.

Line 4: turns off the flag that lets other procedures know whether they can introduce new quantities. This prevents equations being formed which introduce new quantities.

Line 5: the 'chooseeqn' predicate is given 'X', a sought from the list of soughts, the list of types of quantities, and 'Us', the list of used plans which will initially be empty. It works out 'Ez', an equation containing the quantity bound to 'X', and no other unknown quantity. The plan used to generate this equation is

put in 'U'. If no such equation can be generated, this (second) 'geteqns', clause will fail and control will pass to the third 'geteqns' clause on line 13, which will allow new unknowns to be introduced.

Line 6: this simplifies equation 'Ez' and places it in 'E'. The PROLOG pattern-matcher matches the output of this procedure with the first element of the conjunction in Argument 5 of the head goal. Thus the simplified equation is added to the conjunction of equations being formed by the procedure.

Line 7: the quantities appearing in equation 'E' are put in a list 'Ws'.

Line 8: this checks that the quantity in 'X' for which the equation was generated actually appears in the list 'Ws'.

Lines 9 & 10: these two procedures form a list 'Ys' consisting of all quantities currently sought and given. The quantities in the current equation are checked against this list, and if the equation contains quantities not in the list, the 'subtract' procedure will fail. This

will cause the program to backtrack to line 5 and attempt to produce another equation.

Line 11: this prevents the choices made already from being changed by backtracking: if an equation has been generated and passed the tests in lines 8-11, then it must be correct.

Line 12: 'geteqns' is called recursively to generate equations to solve for the remaining unknowns (Argument 1), having added the plan 'U' to the list of plans used (Argument 4).

The Third Clause:

Line 13: the head goal of the second 'geteqns' clause. This clause will allow for the introduction of intermediate unknowns.

Line 14: separates the head goal and the body of the clause.

Line 15: turns on the flag that allows creation of intermediate unknowns.

Lines 16-20: are the same as lines 5-9. The 'chooseeqn' predicate is now enabled to introduce new quantities, because the flag has been set at line 15.

Line 21: 'Zs' is instantiated to a list of quantities introduced by the new equation 'E'.

Line 22: the new quantities introduced are added to 'Xs', the old list of soughts, to give 'NXs', the new list of soughts.

Line 23: this gets the types of the new quantities and adds them to the list of types to give 'N_types'.

Line 24: 'geteqns' is now called recursively to find equations to solve for the remaining quantities.

Since the actual formation of the equation is done by the 'chooseeqn' predicate, the code for this has been included as well. 'Chooseeqn' has five arguments:

Argument 1: the quantity that an equation is needed to solve for. This will be input when the procedure is called.

Argument 2: a list of the types of all sought or given quantities. This will be input when the procedure is called.

Argument 3: the equation produced. This will be output by the procedure.

Argument 4: the plan used to generate the equation. This is output by the procedure.

Argument 5: a list of plans already used. This is input when the procedure is called, and is used to check that the same plan is not used twice. (Otherwise non-independent equations will result).

There is only one procedure for 'chooseeqns' given in lines 25-32 above.

Line 25: the head goal of the procedure. When this is called, Arguments 1, 2 and 5 will be instantiated and arguments 3 and 4 will be supplied by the procedure as output.

Line 27: the input to this procedure is the sought quantity 'Q'. The procedure finds the type of 'Q' and puts it

in 'Qtype'. 'Pred' is the name of a predicate in the problem representation which contains the sought quantity and 'Args' is a list of its arguments. This information will be used in identifying a context in which to apply the selected formula.

Line 28: constructs a list of physical principles that relate quantities of type 'Qtype' to other quantities in the list 'Types'. This list is output as 'Formulae-list'

Line 29: selects a member of the list 'Formulae-list', and calls it 'Formula'. This selection can be re-done if necessary by the backtracking mechanism.

Line 30: forms a plan called ' Strategy' consisting of the chosen formula, and a context in which to apply it. The structure of the context is defined by the nature of the formula and will involve an object of a defined type. The objects of the appropriate type mentioned in the problem representation are scanned.

An object is selected which is related to the sought quantity, appearing in the list of arguments 'Args' derived from the predicate which introduced the sought quantity.

The rest of the context is then supplied by reference to the problem representation and the list of arguments to the predicate introducing the sought quantity. This process differs for different types of context in which different physical principles can be applied.

Line 31: checks that the plan called 'Strategy' is different from previously used plans.

Line 32: takes the plan as input and constructs an equation from it. There are 'makeqn' clauses for every possible combination of principle and context type.

This is the full implementation of the backward inference problem-solving strategy used by NEWT. It defines the order of equation generation, and the introduction of new quantities. This forms the core of the NEWT system, and underlies the frameworks for correct problem-solving and consistently erroneous problem-solving.

The correct and incorrect equation generating procedures are invoked by the "makeqn" predicate. These will not be given in

detail, since their internal structure within NEWT is not claimed to be psychologically representative.

Appendix II

Experimental Data

Since NEWT is only concerned with output of equations, the data are presented in a simplified form, with only the equations put down.

Since a repeated answer counts in this analysis as equivalent to the repetition of the equations, cases where repeat answers were given will be represented by a repeat set of equations. Lines of working derivable from previous equations are not included.

Question 1 - The Slope Question

Question 2 - The Strut Question

Question 3 - The Angle of Friction Question

Question 4 - The Unsliding Block Question

The questions are labelled Q1V1 for the first version of Question 1, Q1V2 for the second version and so on.

Student	Question	Equations Produced
S1	Q1V1	R - W * g * cos theta = 0
		F - W * g * sin theta = 0
		(theta is marked as the angle between the vertical and the normal)
	Q1V2	R = 32 * cos theta
		F = 32 * sin theta
	Q2V1	R - m * g + T * sin 40 = 0
		m * g * L - R3 * L - T1 * sin 40 = 0
		R4 = 0
		(R3 is the vertical force on the end of the vertical string due to the strut alone. T1 is the tension in the 40 degree string, and R4 is the force of compression in the strut)

	Q2V2	no equation
	Q3V1	F = mu * R R - 6 * g * cos 40 = 0 F - m * g * sin 40 = 0
	Q3V2	remembered answer counts as same equations
	Q4V1	no equation
	Q4V2	no equation
S2	Q1V1	R = 3.2 * 10 * cos 35 F = 3.2 * g * sin 35
	Q1V2	R = 3.2 * g * cos 35 W = 3.2 * g * sin 35 (since W is evaluated and presented as an answer, it is not clear whether it is intended to refer to the weight or the force of friction)

Q2V1 $T = \sin 40 * m * g$

Q2V2 $T = \sin 40 * m * g$

Q3V1 $F = 60 * \cos 40$

$P = 60 * \sin 40$

Coefficient of Friction $= \frac{F}{P}$

Q3V2 Coefficient of Friction $= \frac{60 * N \cos 40}{60 * N \sin 40}$

Q4V1 $mu * b = \sin 45 * P$

$\sin 45 * P = a * g$

(P is not defined)

Q4V2 $mu * b = P * \sin 45$

$\sin 45 = \frac{a * g}{P}$

(P is marked as a vector quantity collinear with the string at 45 degrees to the horizontal)

S3 Q1V1 $W = 3.2 * 10$

$\cos 35 = \frac{35}{RW}$

(RW is not defined)

Q1V2 no equation

Q2V1 $\sin 40 = \frac{m * g}{T}$

Q2V2 no equation

Q3V1 FF = FS

(FF is shown parallel to the plane and moving up, FS is shown parallel to the plane moving down)

Q3V2 $X = \frac{6}{\cos 50}$

(X is shown as a vector parallel to the plane and moving down)

	Q4V1	no equation
	Q4V2	no equation
S4	Q1V1	$F = mu * R$ $\frac{F}{mu} = \frac{m * g * \cos 35}{\tan 35}$
	Q1V2	$F = mu * R$ $\frac{F}{mu} = \frac{m * g * \cos \text{theta}}{\tan \text{theta}}$
	Q2V1	$T = m * g * \sin 40$
	Q2V2	$T = m * \cos 40$
	Q3V1	$F = mu * R$ $\frac{F}{R} = \frac{m * g * \cos 40}{m * g * \sin 40}$
	Q3V2	$mu = \tan 40$
	Q4V1	no equation

	Q4V2	no equation
S5	Q1V1	$F = m * A$
		$r = \frac{w * F * m}{W}$
		35 degrees $= \frac{F * m}{w * r}$
		$\frac{35}{3.2} = F$
		$F = 112$
		$F = m * a * \cos 35$
		$F = 3.2 * 9.8 * \cos 35$
		(there is no definition of m, r, w, A or a)
	Q1V2	A numerical value (26.2) is given for F which does not equal any of the previously obtained values (112, 25.69 or 17.98). This is therefore not considered as equivalent to a restatement of any previous equation.

Q2V1 $T = m * \text{theta}$

(there is no definition of theta)

Q2V2 no equation

Q3V1 no equation

Q3V2 no equation

Q4V1 $\frac{mu}{P} = \cos \text{theta} * a * p$

$$a = \frac{mu}{\cos \text{theta} * P}$$

$$b = \sin \text{theta} * P$$

$$b = \frac{a * mu}{\sin \text{theta} * \cos \text{theta} * P}$$

$$a = \frac{b * mu}{\sin \text{theta} * \cos \text{theta} * P}$$

(there is no definition of P or theta)

Q4V2 $a = \dfrac{mu}{b * \sin\text{ theta} * \cos\text{ theta}}$

(theta is not defined)

S6 Q1V1 $F - 3.2 * g * \sin 35 = 3.2 * 0$

$R - 3.2 * g * \cos 35 = 0$

Q1V2 $F - 3.2 * g * \sin 35 = 0$

$R - 3.2 * g * \cos 35 = 0$

Q2V1 $T * \sin 40 - m * g = 0$

Q2V2 $T * \sin 40 - m * g = 0$

Q3V1 $F - 6 * g * \sin 40 = 0$

$R - 6 * g * \cos 40 = 0$

$F = mu * R$

Q3V2 $\tan 40 = 0.8390996$

$mu = 0.84$

(Taken as: mu =tan 40)

S6	Q4V1	$F = mu * R$ $a * g * \cos 45 = F$ $a = \frac{mu * b}{g * \cos 45}$
	Q4V2	$a = \frac{b * mu}{g * \cos 45}$
S7	Q1V1	$F = \cos 35 * W$
	Q1V2	$F = \cos 35 * W$
	Q2V1	no equation
	Q2V2	no equation
	Q3V1	no equation
	Q3V2	no equation
	Q4V1	no equation
	Q4V2	no equation

S8 Q1V1 $R = m * g * \cos 35$

$F = m * g * \sin 35$

Q1V2 $R = 10 * 3.2 * \cos 35$

$F = 10 * 3.2 * \sin 35$

Q2V1 $\frac{T}{\sin 40} = m * g$

Q2V2 $T = m * 10 * \sin 40$

Q3V1 $R = m * g * \cos \theta$

$F = m * g * \sin \theta$

$\mu = F/R$

(theta is clearly identified with the 40 degree angle, and m with the 6kg mass)

Q3V2 $R = m * g * \cos 40$

$F = m * g * \sin \theta$

$F = \mu * R$

Q4V1 no equation

Q4V2 $R = b * g$

$F = \mu * b * g$

$$\frac{\mu * b * g}{\sin 135} = \frac{a * g}{\sin 135}$$

S9 Q1V1 $R = W * \cos 35$

$F = mg \sin 35$

Q1V2 $R = 3.2 * 9.8 * \cos 35$

$F = 3.2 * 9.8 * \sin 35$

Q2V1 no equation

Q2V2 no equation

Q3V1 $R = 6 * 9.8 * \cos 40$

$Fr = 6 * g * \sin 40$

$Fr = mu * R$

Q4V1 $T = F$

$F = mu * R$

$mu * R = mu * b * g$

$$\frac{a * g}{\sin 135} = \frac{mu * b * g}{\sin 135}$$

Q4V2 $R = b * g$

$T = F$

$F = mu * R$

$$\frac{mu * b * g}{\sin 135} = \frac{a * g}{\sin 135}$$

S10 Q1V1 $F - 3.2 * g * \sin 35 = 3.2 * 0$

$R - 3.2 * g * \cos 35 = 0$

Q1V2 $F - 3.2 * g * \sin 35 = 0$

$R - 3.2 * g * \cos 35 = 0$

Q2V1 $T * \sin 40 - m * g = 0$

Q2V2 $T * \sin 40 - m * g = 0$

Q3V1 $F - 6 * g * \sin 40 = 0$

$R - 6 * g * \cos 40 = 0$

$F = mu * R$

Q3V2 $F - 6 * g * \sin 40 = 0$

$R - 6 * g * \cos 40 = 0$

$F = mu * R$

Q4V1 $F = a * \cos 45$

$R - b * g = 0$

$F = mu * R$

(F is marked as the tension in the horizontal string, and R as the vertical reaction between b and the table)

Q4V2 $$a = \frac{b * g * mu}{\cos 45}$$

(This is the same result as before, and hence is counted as equivalent to the same equations)

S11	Q1V1	R = 3.2 * 10 * cos 35
		F = 3.2 * 10 * sin 35
	Q1V2	F = 3.2 * 10 * sin 35
		R = 3.2 * 10 * cos 35
	Q2V1	T = m * g * sin 40
	Q2V2	T = m * g * sin 40
	Q3V1	no equation
	Q3V2	no equation
	Q4V1	no equation
	Q4V2	no equation
S12	Q1V1	R = m * g * cos 35
		F = m * g * sin 35
	Q1V2	R = 3.2 * g * cos 35
		F = 3.2 * g * sin 35

Q2V1 no equation

Q2V2 no equation

Q3V1
$$R = 6 * g * \cos 40$$
$$F = 6 * g * \sin 40$$
$$mu = F / R$$

Q3V2
$$R = 6 * g * \cos 40$$
$$F = 6 * g * \sin 40$$
$$mu = F / R$$

Q4V1
$$T = F$$
$$F = mu * R$$
$$R = b * g$$
$$\frac{a * g}{\sin 135} = \frac{b * g * mu}{\sin 135}$$

(T is given as the tension in the horizontal string)

Q4V2
$$\frac{a * g}{\sin 135} = \frac{mu * b * g}{\sin 135}$$

S13	Q1V1	R - W * cos 35 = 0
		W * sin 35 = F
	Q1V2	R = W * cos 35
		F = W * sin 35
	Q2V1	sin 40 = m * g / T
	Q2V2	T = m * g * sin 40
	Q3V1	R = 60 * cos 40
		60 * sin 40 = F
		mu = F / R
	Q3V2	F = mu * R
		F = 60 * sin 40
		R = 60 * cos 40
	Q4V1	no equation
	Q4V2	no equation

S14 Q1V1 $F' = F * \cos \text{theta}$

$$R^2 = W^2 - F^2$$

(where F' is defined as the force down the slope i.e, the friction force, and F is defined as the weight)

Q1V2 $W = m * g$

$F = W * \cos \text{theta}$

(here, F is taken as the friction force)

Q2V1 $T = m * \cos 50$

Q2V2 $T = m * g * \cos 40$

Q3V1 $\text{Force} = m * g$

$\text{Friction Force} = F * \cos \text{theta}$

(theta is identified as 40 degrees, and F as "Force")

Q3V2

$F = m * g$

$Ff = F * \cos \text{theta}$

(Ff is identified as the friction force)

Q4V1

Force along a - > wall = $a * g * \cos 45$

Force along P - > b = $a * g * \sin 90$

at b force = $b * g$

due to friction = $a * g * \sin 90$

$$a = \frac{b * g * mu * \cos 45}{\sin 90}$$

Q4V2

$a = m * g$

$F1 = mu$

$b = m * g$

$a = b * mu * \cos 45$

(m is not defined)

S15 Q1V1

$F - 3.2 * g * \sin 35 = 0$

$N - 3.2 * g * \cos 35 = 0$

(N is defined as the normal reaction between the body and the plane)

Q1V2

$F - 3.2 * g * \sin 35 = 0$

$R - 3.2 * g * \cos 35 = 0$

Q2V1

$$T = \frac{m * g}{\sin 40}$$

Q2V2

no equation

Q3V1

$F - \sin 40 * 6 * g = 0$

$R - \cos 40 * 6 * g = 0$

$F = mu * R$

Q3V2

$F - 6 * g * \sin 40 = 0$

$R - 6 * g * \cos 40 = 0$

$mu = F / R$

Q4V1

no equation

	Q4V2	no equation
S16	Q1V1	F - W * sin 35 = 0
		R - W * cos 35 = 0
	Q1V2	F - W * sin 35 * g = 0
		R - 3.2 * g * cos 35 = 0
	Q2V1	no equation
	Q2V2	no equation
	Q3V1	F - 6 * g * sin 40 = 0
		R - 6 * g * cos 40 = 0
		F = mu * R
	Q3V2	F - 6 * sin 40 * g = 0
		R - 6 * cos 40 * g = 0
		F = mu * R
	Q4V1	no equation
	Q4V2	no equation

S17	Q1V1	R - 3.2 * g * cos 35 = 0
	Q1V2	R - 3.2 * g * cos 35 = 0 F = 3.2 * g * sin 35
	Q2V1	no equation
	Q2V2	no equation
	Q3V1	R - 6 * g * cos 40 = 0 F - 6 * g * sin 40 = 0 F = mu * R
	Q3V2	R - 6 * g * cos 40 = 0 F - 6 * g * sin 40 = 0 F = mu * R
	Q4V1	no equation
	Q4V2	no equation
S18	Q1V1	R - W * cos 35 = 0 F - W * sin 35 = 0

Q1V2 $R - W * \cos 35 = 0$

$F - W * \sin 35 = 0$

Q2V1 $T = \frac{m * g}{\sin 40}$

Q2V2 $T = \frac{m * g}{\sin 40}$

Q3V1 $R - 6 * g * \cos 40 = 0$

$F - 6 * g * \sin 40 = 0$

$F = mu * R$

Q3V2 $R - 6 * g * \cos 40 = 0$

$F - 6 * g * \sin 40 = 0$

$F = mu * R$

Q4V1 $F = mu * R$

(F is defined as the surface friction between b and the table, and R as the normal reaction between them)

Q4V1 no equation

	Q4V2	no equation
S19	Q1V1	R - 3.2 * 9.8 * cos 35 = 0
	Q1V2	R - 3.2 * g * cos 35 = 0 F - 3.2 * g * sin 35 = 0 F = mu * R
	Q2V1	no equation
	Q2V2	no equation
	Q3V1	R - 6 * g * cos 40 = 0 F - 6 * g * sin 40 = 0 38.56 = mu * 45.96 (F is evaluated to 38.56 and R is evaluated to 45.96. This is therefore equivalent to F = mu * R)
	Q3V2	R - 6 * g * cos 40 = 0 F - 6 * g * sin 40 = 0 F = mu * r

	Q4V1	no equation
	Q4V2	no equation
S20	Q1V1	$R = 3.2 * g * \cos 35$ $F - 3.2 * g * \sin 35 = 0$
	Q1V2	$R = 3.2 * g * \cos 35$ $F - 3.2 * g * \sin 35 = 0$
	Q2V1	$m * g - T * \sin 40 = 0$
	Q2V2	$T = \frac{m * g}{\sin 40}$
	Q3V1	$R = 6 * g * \cos 40$ $mu * R - 6 * g * \sin 40 = 0$
	Q3V1	$R = 6 * g * \cos 40$ $mu * R - 6 * g * \sin 40 = 0$

Q4V1 $T1 * \cos 45 - T1 - mu * R = 0$

$R = b * g$

$b * g + a * g - T1 * \sin 45 = 0$

(T1 is marked as being the tension in both the horizontal and inclined strings)

Q4V2 no equation

S21 Q1V1 $R = 3.2 * g * \cos 35$

$F = W * \sin 35 * g$

Q1V2 $R = 3.2 * g * \cos 35$

$F = 3.2 * g * \sin 35$

Q2V1 $$T = \frac{m * g}{\sin 40}$$

Q2V2 $T * \sin 40 = m * g$

Q3V1

F = 6 * g * sin 40

R = 6 * g * cos 40

F = mu * R

Q3V2

R = 6 * g cos 40

F = 6 * g sin 40

37.795 = mu * 45.64

(F is evaluated to 37.795,and R is evaluated to 45.64. This is therefore equivalent to F = mu * R)

Q4V1

T * sin 45 = a * g

T * cos 45 = mu

(T is marked as the tension in the inclined string)

Q4V2

T1 * cos 45 = b * mu

a * g = sin 45 * T1

(T1 is marked as the tension in the inclined string)

S22 Q1V1

$R = W * \cos 35$

$F = W * \sin 35$

Q1V2

$3.2 * g * \sin 35 = F$

$3.2 * g * \cos 35 = R$

Q2V1

$$\frac{R}{\sin 130} = \frac{T}{\sin 90} = \frac{m * g}{\sin 140}$$

(R is defined as the compression in the strut)

Q2V2

$$\frac{m * g}{\sin 140} = \frac{R}{\sin 130} = \frac{T}{\sin 90}$$

Q3V1

$F = mu * N$

$6 * g * \sin 40 = mu * N$

$6 * g * \cos 40 = N$

(N is defined as the normal reaction)

Q3V2

$6 * g * \cos 40 = R$

$6 * g * \sin 40 = mu * R$

Q4V1

$R = b * g$

$mu * R = T3$

$$\frac{a * g}{\sin 135} = \frac{T1}{\sin 90} = \frac{T3}{\sin 135}$$

(T3 is the tension in the horizontal string, and T1 is the tension in the inclined string)

Q4V2

$R = b * g$

$T3 = mu * R$

$$\frac{a * g}{\sin 135} = \frac{T3}{\sin 135} = \frac{T2}{\sin 90}$$

S23 Q1V1

$F - 3.2 * g * \sin 35 = 0$

$R - 3.2 * g * \cos 35 = 0$

Q1V2

$F - W * \sin 35 = 0$

$R - W * \cos 35 = 0$

Q2V1

no equation

Q2V2

no equation

Q3V1	no equation
Q3V2	F - 6 * g * sin 40 = 0
	R - 6 * g * cos 40 = 0
	F = mu * R
Q4V1	no equation
Q4V2	no equation

Appendix III - Sample NEWT Traces

A sample of NEWT output is given below, showing solutions to the Strut Problem and the Unsliding Block problem. Since these printouts were made, the malrules in NEWT have been renumbered for greater consistency. The malrule numbers mentioned in the trace output therefore do not necessarily correspond to those in the headings.

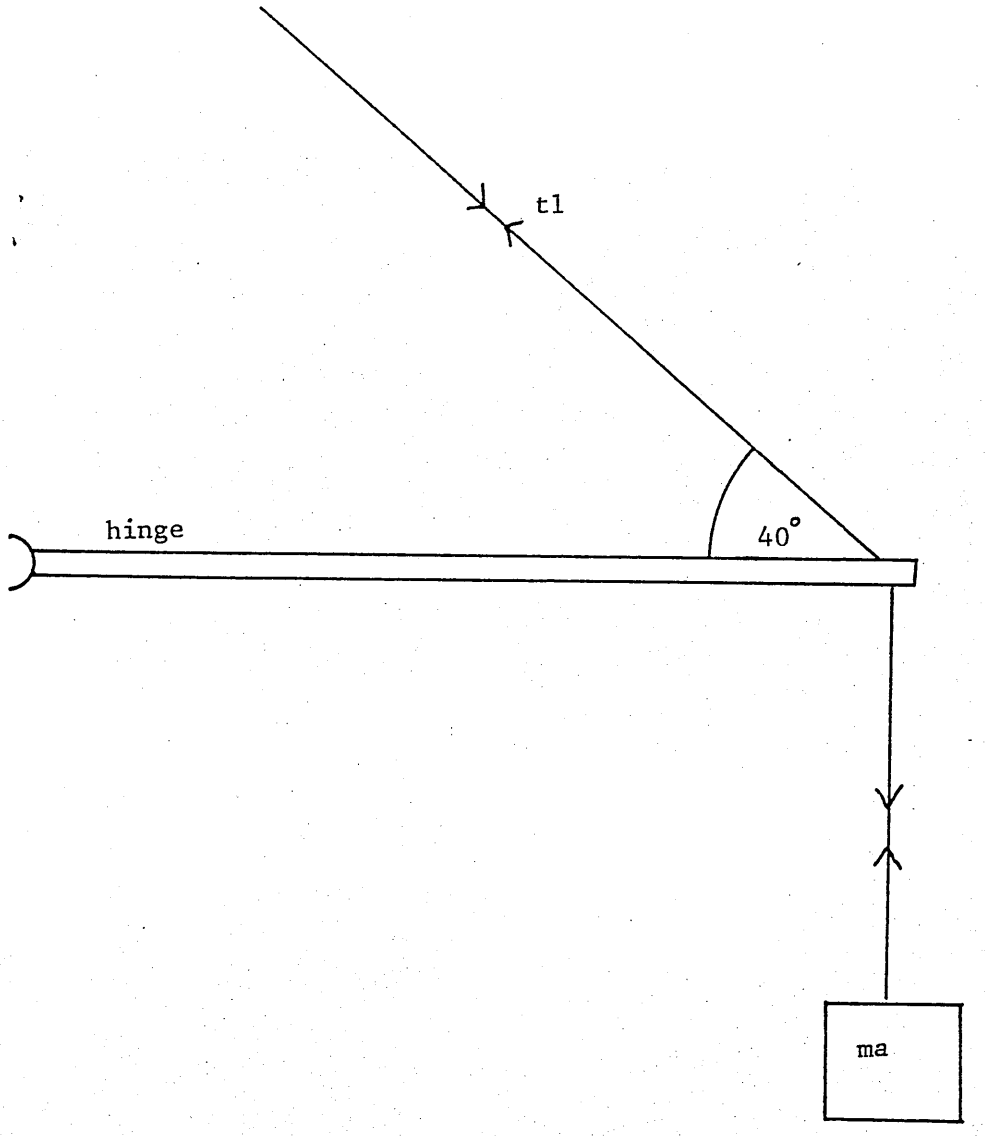


Figure 1. The Strut Problem

The Strut Problem

Correct Solution

```
so.

ttempting to solve for *t1* in terms of *ma*
 am now trying to solve for t1 without introducing any unknowns.
 luck - ) will now accept unknowns in solving for t1.
et tension1 be the tension of string2.

quation- 1: -tension1*cos(50)+t1=0
rmed by resolving forces at 140 to the horizontal for strut

his equation solves for t1 but introduces *tension1*.
So now I must solve for *tension1*,
          given *t1,ma*
 am now trying to solve for tension1 without introducing any unknowns.
quation- 2: tension1-ma*g=0
rmed by resolving forces at 90 to the horizontal for a

his equation solves for tension1.

quations extracted :
   -tension1*cos(50)+t1=0
   tension1-ma*g=0


s
```

The Strut Problem

Solved using Malrule11

C = R * cos theta

```
    assert(ma!rule)7).

es
 ?- so.

ttempting to solve for ↑t↓ in terms of ↑ma↓
 am now trying to solve for t1 without introducing any unknowns.
o luck - I will now accept unknowns in solving for t1.
Let tension1 be the tension of string2.


quation- 1: tension1*cos(theta)+t=0
ormed by resolving forces at 140 degrees to the horizontal for strut

This equation solves for t1 but introduces ↑tension1,theta↓.
* ERROR Type unknown -- ↑theta↓
  ( continue after error )

To find ↑tension1,theta↓,   given ↑t1,ma↓
 am now trying to solve for tension1 without introducing any unknowns.

quation- 2: tension1-ma*g=0
ormed by resolving forces at 90 degrees to the horizontal for a

This equation solves for tension1.
To find ↑theta↓,  given ↑tension1,t1,ma↓
 am now trying to solve for theta without introducing any unknowns.
o luck - I will now accept unknowns in solving for theta.

 am unable to solve for theta.

 will go back to solve for t1 again

Equation-1 rejected.


 am unable to solve for t1.

 ?-
```

The Strut Problem

Solved using Malrule1

Weight = Mass

```
*ma1rule19.

yes
| ?- go.

Attempting to solve for ↑t1← in terms of ↑ma←

I am now trying to solve for t1 without introducing any unknowns.

 Applicable formulae : ↑moments,resolve←

No luck - I will now accept unknowns in solving for t1.

 Applicable formulae : ↑moments,resolve←
 Let tension1 be the tension of string2.
 Note: tension1 (of type force) was used in a tension definition (2)

 Equation-1 : -tension1*cos(50)+t1=0
 formed by applying : strategy(resolve,situation(rigid1),strut,↑typical_
point1,rend,lend←,140,now))

 This equation solves for t1 but introduces ↑tension1←.

 So now I must solve for ↑tension1←
                  given ↑t1,ma←

I am now trying to solve for tension1 without introducing any unknowns.

 Applicable formulae : ↑moments,resolve←

 Equation-2 : tension1+(-ma)=0
 formed by applying : strategy(resolve,situation(particle,a,↑bottom2,a←,
90,now))

 This equation solves for tension1.

 So now I must solve for ↑←
                  given ↑tension1,t1,ma←


Equations extracted :
    -tension1*cos(50)+t1=0
    tension1+(-ma)=0


yes
| ?-
```

The Strut Problem

Solved using Malrule16

C = R * sign (cos a)

```
    malrule41.

:s
?- eq.

tempting to solve for ↑ti↑ in terms of ↑ma↑

am now trying to solve for t1 without introducing any unknowns.

luck - I will now accept unknowns in solving for t1.
et tension1 be the tension of strine2.

uation- 1: -tension1+t1=0
rmed by resolving forces at 140 to the horizontal for strut

his equation solves for t1 but introduces ↑tension1↑.

o now I must solve for ↑tension1↑,          given ↑t1,ma↑

am now trying to solve for tension1 without introducing any unknowns.

uation- 2: tension1-ma*g=0
rmed by resolving forces at 90 to the horizontal for a

his equation solves for tension1.

uations extracted !
  -tension1+t1=0
  tension1-ma*g=0

s
```

The Strut Problem

Solved using Malrule17

C = R * sin a

```
?-
    assert(na1rule44).

es
 ?- so.

ttempting to solve for ↑t1↑ in terms of ↑ma↑
 am now trying to solve for t1 without introducing any unknowns.
o luck - I will now accept unknowns in solving for t1.
Let tension1 be the tension of string2.

quation- 1: tension1*sin(-130)+t1=0
ormed by resolving forces at 140 to the horizontal for strut

This equation solves for t1 but introduces ↑tension1↑.
So now I must solve for ↑tension1↑,
          given ↑t1,ma↑
 am now trying to solve for tension1 without introducing any unknowns.
quation- 2: tension1-ma*g=0
ormed by resolving forces at 90 to the horizontal for a

This equation solves for tension1.

quations extracted :
   tension1*sin(-130)+t1=0
   tension1-ma*g=0


es
```

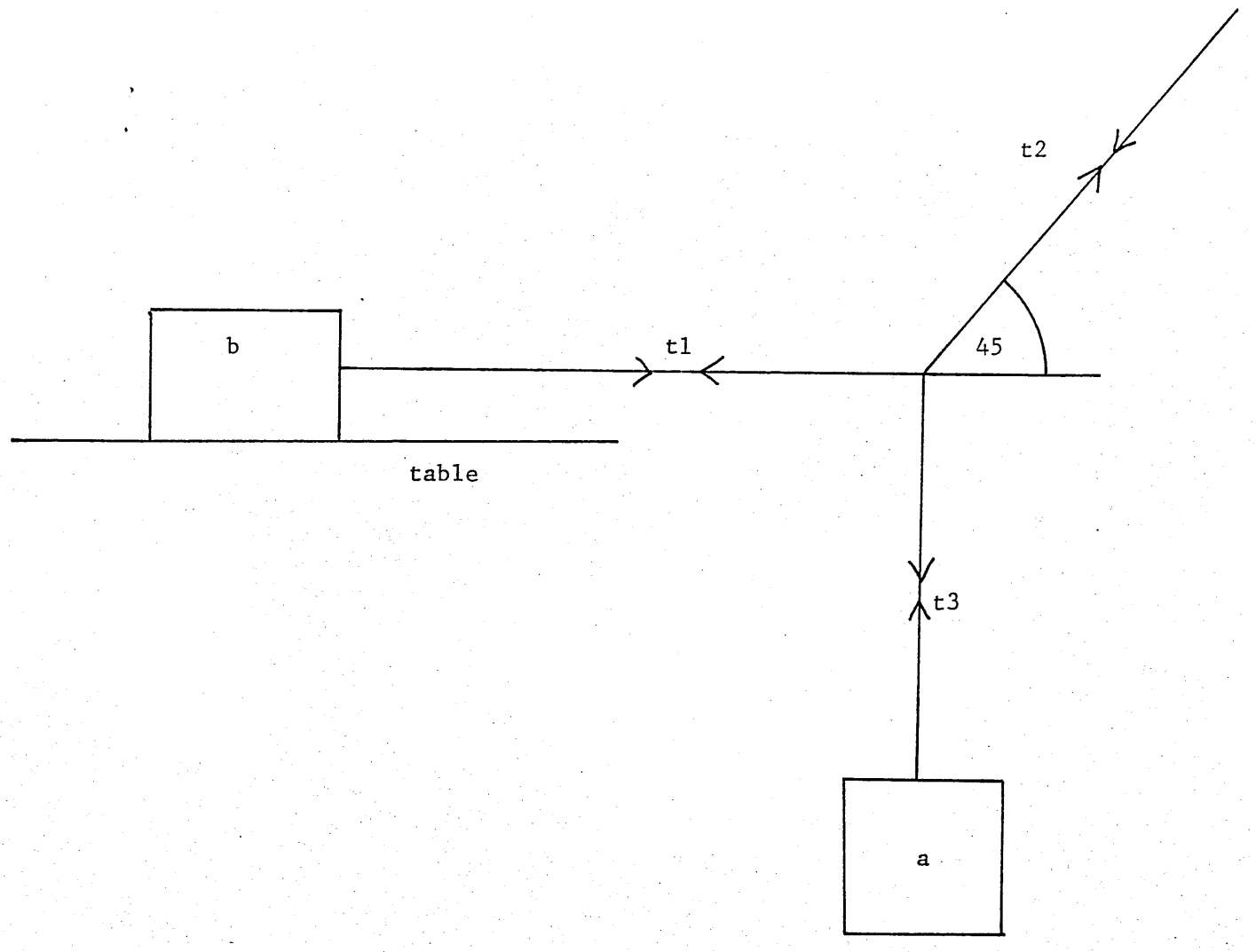


Figure 2. The Unsliding Block Problem

The Unsliding Block Problem

Correct solution

[illegible] for ^ma^ in terms of ^mb,mu^

am now trying to solve for ma without introducing any unknowns.

pplicable formulae : ^resolve^

uck - I will now accept unknowns in solving for ma.

pplicable formulae : ^resolve^
et tension1 be the tension of string3.
ote: tension1 (of type force) was used in a tension definition (2)

quation-1 : ma*g*cos(-90)+tension1=0
ormed by applying : strategy(resolve,situation(particle,a,^a,pt5^,92,
)

his equation solves for ma but introduces ^tension1^.

o now I must solve for ^tension1^
given ^ma,mb,mu^

am now trying to solve for tension1 without introducing any unknowns.

pplicable formulae : ^moments,resolve^

luck - I will now accept unknowns in solving for tension1.

pplicable formulae : ^moments,resolve^
et tension2 be the tension of string2.
ote: tension2 (of type force) was used in a tension definition (2)
et tension3 be the tension of string1.
ote: tension3 (of type force) was used in a tension definition (2)

(ation-2 : tension1*cos(-180)+(tension2*cos(45)+tension3*cos(-90))=0
ormed by applying : strategy(resolve,situation(nrps,pt5,^pt5,pt3,pt2^,
,now))

his equation solves for tension1 but introduces ^tension2,tension3^.

o now I must solve for ^tension2,tension3^
given ^tension1,ma,mb,mu^

am now trying to solve for tension2 without introducing any unknowns.

pplicable formulae : ^moments,resolve^

quation-3 : tension1*cos(-45)+(tension2* -1+tension3*cos(45))=0
ormed by applying : strategy(resolve,situation(nrps,pt5,^pt5,pt3,pt2^,
5,now))

his equation solves for tension2.

```
am now trying to solve for tension3 without introducing any unknowns.

pplicable formulae : *moments.resolve*

, luck - I will now accept unknowns in solving for tension3.

pplicable formulae : *moments.resolve*
et reaction1 be the reaction of table in direction 90.
ote: reaction1 (of type force) was used in a reaction definition (3)

 ation-4 : tension3* -1+(mu*reaction1+mb*g*cos(-90))=0
ormed by applying : strategy(resolve,situation(particle,b,*pt1,pt,b*,1
),now))

his equation solves for tension3 but introduces *reaction1*.

o now I must solve for *reaction1*
                given *tension3-tension2,tension1,ma,mb,mu*

am now trying to solve for reaction1 without introducing any unknowns.

pplicable formulae : *moments.resolve*

quation-5 : reaction1+mu*reaction1*cos(-90)+mb*g*cos(-180)=0
ormed by applying : strategy(resolve,situation(particle,b,*pt1,pt,b*,90
now))

his equation solves for reaction1.

 now I must solve for **
                given *reaction1,tension3-tension2,tension1,ma,mb,mu*


uations extracted :
   ma*g*cos(-180)+tension1=0
   tension1*cos(-180)+(tension2*cos(45)+tension3*cos(-90))=0
   tension1*cos(-45)+(tension2* -1+tension3*cos(45))=0
   tension3* -1+(mu*reaction1+mb*g*cos(-90))=0
   reaction1+mu*reaction1*cos(-90)+mb*g*cos(-180)=0
```

The Unsliding Block Problem

Solved using Malrule12

C = R * a

```
    assert(ma(rule=2)).

es
 ?- go.

tement is to solve for *ma* in terms of *mb,mu*
 am now trying to solve for ma without introducing any unknowns.
o luck - I will now accept unknowns in solving for ma.
Let tension1 be the tension of string5.


quation- 1: -ma*g+tension1=0
ormed by resolving forces at 90 degrees to the horizontal for a

This equation solves for ma but introduces *tension1*.
To find *tension1*,   given *ma,mb,mu*
 am now trying to solve for tension1 without introducing any unknowns.
o luck - I will now accept unknowns in solving for tension1.
Let tension2 be the tension of string2.
Let tension3 be the tension of string1.


quation- 2: -tension1+tension2*45=0
ormed by resolving forces at 90 degrees to the horizontal for pt5

This equation solves for tension1 but introduces *tension2,tension3*.
To find *tension2,tension3*,   given *tension1,ma,mb,mu*
 am now trying to solve for tension2 without introducing any unknowns.

quation- 3: tension1*45-tension2+tension3*45=0
ormed by resolving forces at 225 degrees to the horizontal for pt5

This equation solves for tension2.
To find *tension3*,   given *tension2,tension1,ma,mb,mu*
 am now trying to solve for tension3 without introducing any unknowns.
o luck - I will now accept unknowns in solving for tension3.
Let reaction1 be the reaction of table in direction 90.


quation- 4: -tension3+mu*reaction1=0
ormed by resolving forces at 180 degrees to the horizontal for b

This equation solves for tension3 but introduces *reaction1*.
To find *reaction1*,   given *tension3,tension2,tension1,ma,mb,mu*
I am now trying to solve for reaction1 without introducing any unknowns.

quation- 5: reaction1-mb*g=0
ormed by resolving forces at 90 degrees to the horizontal for b

This equation solves for reaction1.

quations extracted :
    -ma*g+tension1=0
    -tension1+tension2*45=0
    tension1*45-tension2+tension3*45=0
    -tension3+mu*reaction1=0
    reaction1-mb*g=0
```

The Unsliding Block Problem

Solved using Malrule11

C = R * cos theta

```
yes
| ?- go.

Attempting to solve for *ma* in terms of *mb mu*
I am now trying to solve for ma without introducing any unknowns.
No luck - I will now accept unknowns in solving for ma.
 Let tension1 be the tension of string3.


Equation- 1: -ma*g+tension1=0
Formed by resolving forces at 90 degrees to the horizontal for a

 This equation solves for ma but introduces *tension1*.
 To find *tension1*,   given *ma,mb,mu*
I am now trying to solve for tension1 without introducing any unknowns.
No luck - I will now accept unknowns in solving for tension1.
 Let tension2 be the tension of string2.
 Let tension3 be the tension of string1.


Equation- 2: -tension1+tension2*cos(theta)=0
Formed by resolving forces at 90 degrees to the horizontal for pt5

 This equation solves for tension1 but introduces *tension2,theta,tensio
n3*.
** ERROR Type unknown -- *theta*
    ( continue after error )

 To find *tension2,theta,tension3*,   given *tension1,ma,mb,mu*
I am now trying to solve for tension2 without introducing any unknowns.

Equation- 3: tension1*cos(theta)+(-tension2+tension3*cos(theta))=0
Formed by resolving forces at 225 degrees to the horizontal for pt5

 This equation solves for tension2.
 To find *theta,tension3*,   given *tension2,tension1,ma,mb,mu*
I am now trying to solve for theta without introducing any unknowns.
No luck - I will now accept unknowns in solving for theta.

I am unable to solve for theta.

I will go back to solve for tension1 again

 Equation-2 rejected.


I am unable to solve for tension1.

I will go back to solve for ma again

 Equation-1 rejected.


Equation- 4: 0=0
Formed by resolving forces at 0 degrees to the horizontal for a

 This equation solves for ma but introduces *tension1*.
 To find *tension1*,   given *ma,mb,mu*
I am now trying to solve for tension1 without introducing any unknowns.

Equation- 5: -ma*g+tension1=0
Formed by resolving forces at 90 degrees to the horizontal for a

 This equation solves for tension1.

Equations extracted :
```

The Unsliding Block Problem

Solved using Malrule16

C = R * sign (cos a)

yes
| ?- go.

Attempt to solve for <ma> in terms of <mb,mu>

I am now trying to solve for ma without introducing any unknowns.

No luck - I will now accept unknowns in solving for ma.
Let tension1 be the tension of string3.

Equation- 1: -ma*g+tension1=0
Formed by resolving forces at 90 to the horizontal for a

This equation solves for ma but introduces ↑tension1↓.

So now I must solve for ↑tension1↓, given ↑ma,mb,mu↓

I am now trying to solve for tension1 without introducing any unknowns.

No luck - I will now accept unknowns in solving for tension1.
Let tension2 be the tension of string2.
Let tension3 be the tension of string1.

Equation- 2: -tension1+tension2=0
Formed by resolving forces at 90 to the horizontal for pt5

This equation solves for tension1 but introduces ↑tension2,tension3↓.

So now I must solve for ↑tension2,tension3↓, given ↑tension1,ma,mb
,mu↓

I am now trying to solve for tension2 without introducing any unknowns.

Equation- 3: tension1+(-tension2+tension3)=0
Formed by resolving forces at 225 to the horizontal for pt5

This equation solves for tension2.

So now I must solve for ↑tension3↓, given ↑tension2,tension1,ma,m
b,mu↓

I am now trying to solve for tension3 without introducing any unknowns.

No luck - I will now accept unknowns in solving for tension3.
Let reaction1 be the reaction of table in direction 90.

Equation- 4: -tension3+mu*reaction1=0
Formed by resolving forces at 180 to the horizontal for b

This equation solves for tension3 but introduces ↑reaction1↓.

So now I must solve for ↑reaction1↓, given ↑tension3,tension2,tens
ion1,ma,mb,mu↓

I am now trying to solve for reaction1 without introducing any unknowns.

Equation- 5: reaction1-mb*g=0
Formed by resolving forces at 90 to the horizontal for b

This equation solves for reaction1.

Equations extracted :
-ma*g+tension1=0
-tension1+tension2=0

The Unsliding Block Problem

Solved using Malrule1

Weight = Mass

es
2- so.

ttempting to solve for ↑ma↓ in terms of ↑ma,mu↓

am now trying to solve for ma without introducing any unknowns.

o luck - I will now accept unknowns in solving for ma.
Let tension1 be the tension of string3.

uation- 1: -ma+tension1=0
rmed by resolving forces at 90 to the horizontal for a

his equation solves for ma but introduces ↑tension1↓.

o now I must solve for ↑tension1↓, given ↑ma,mb,mu↓

am now trying to solve for tension1 without introducing any unknowns.

luck - I will now accept unknowns in solving for tension1.
et tension2 be the tension of string2.
et tension3 be the tension of string1.

uation- 2: -tension1+tension2*cos(45)=0
rmed by resolving forces at 90 to the horizontal for pt5

his equation solves for tension1 but introduces ↑tension2,tension3↓.

o now I must solve for ↑tension2,tension3↓, given ↑tension1,ma,mb
u↓

am now trying to solve for tension2 without introducing any unknowns.

uation- 3: tension1*cos(45)+(-tension2+tension3*cos(45))=0
rmed by resolving forces at 225 to the horizontal for pt5

his equation solves for tension2.

o now I must solve for ↑tension3↓, given ↑tension2,tension1,ma,m
nu↓

am now trying to solve for tension3 without introducing any unknowns.

luck - I will now accept unknowns in solving for tension3.
et reaction1 be the reaction of table in direction 90.

uation- 4: -tension3+mu*reaction1=0
rmed by resolving forces at 180 to the horizontal for b

his equation solves for tension3 but introduces ↑reaction1↓.

now I must solve for ↑reaction1↓, given ↑tension3,tension2,tens
1,ma,mb,mu↓

m now trying to solve for reaction1 without introducing any unknowns.

ation- 5: reaction1-mb=0
med by resolving forces at 90 to the horizontal for b

is equation solves for reaction1.

ations extracted :
-ma+tension1=0
-tension1+tension2*cos(45)=0
tension1*cos(45)+(-tension2+tension3*cos(45))=0

The Unsliding Block Problem

Solved using Malrule17

C = R * sin a

```
    assert(malrule44).

es
 ?- go.

ttempting to solve for ↑ma↑ in terms of ↑mb,mu↑
 am now trying to solve for ma without introducing any unknowns.
o luck - I will now accept unknowns in solving for ma.
Let tension1 be the tension of string3.

quation- 1: -ma*g+tension1=0
ormed by resolving forces at 90 to the horizontal for a

This equation solves for ma but introduces ↑tension1↑.
So now I must solve for ↑tension1↑,
          given ↑ma,mb,mu↑
 am now trying to solve for tension1 without introducing any unknowns.
o luck - I will now accept unknowns in solving for tension1.
Let tension2 be the tension of string2.
Let tension3 be the tension of string1.

quation- 2: -tension1+tension2*sin(45)=0
ormed by resolving forces at 90 to the horizontal for pt5

This equation solves for tension1 but introduces ↑tension2,tension3↑.
So now I must solve for ↑tension2,tension3↑,
          given ↑tension1,ma,mb,mu↑
 am now trying to solve for tension2 without introducing any unknowns.
quation- 3: tension1*sin(-45)+(-tension2+tension3*sin(45))=0
ormed by resolving forces at 225 to the horizontal for pt5

This equation solves for tension2.
So now I must solve for ↑tension3↑,
          given ↑tension2,tension1,ma,mb,mu↑
 am now trying to solve for tension3 without introducing any unknowns.
o luck - I will now accept unknowns in solving for tension3.
Let reaction1 be the reaction of table in direction 90.

quation- 4: -tension3+mu*reaction1=0
ormed by resolving forces at 180 to the horizontal for b

This equation solves for tension3 but introduces ↑reaction1↑.
So now I must solve for ↑reaction1↑,
          given ↑tension3,tension2,tension1,ma,mb,mu↑
 am now trying to solve for reaction1 without introducing any unknowns.
quation- 5: reaction1-mb*g=0
ormed by resolving forces at 90 to the horizontal for b

This equation solves for reaction1.

quations extracted :
   -ma*g+tension1=0
   -tension1+tension2*sin(45)=0
   tension1*sin(-45)+(-tension2+tension3*sin(45))=0
   -tension3+mu*reaction1=0
   reaction1-mb*g=0

es
```

The Unsliding Block Problem

Solved using Malrule6

Friction = 0

mpting to solve for ↑ma↓ in terms of ↑mb,mu↓
now trying to solve for ma without introducing any unknowns.
uck - I will now accept unknowns in solving for ma.
tension1 be the tension of string3.

tion- 1: -ma*g+tension1=0
ed by resolving forces at 90 to the horizontal for a

s equation solves for ma but introduces ↑tension1↓.
now I must solve for ↑tension1↓,
given ↑ma,mb,mu↓
now trying to solve for tension1 without introducing any unknowns.
uck - I will now accept unknowns in solving for tension1.
tension2 be the tension of string2.
tension3 be the tension of string1.

tion- 2: -tension1+tension2*cos(45)=0
ed by resolving forces at 90 to the horizontal for pt5

s equation solves for tension1 but introduces ↑tension2,tension3↓.
now I must solve for ↑tension2,tension3↓,
given ↑tension1,ma,mb,mu↓
now trying to solve for tension2 without introducing any unknowns.
tion- 3: tension1*cos(45)+(-tension2+tension3*cos(45))=0
ed by resolving forces at 225 to the horizontal for pt5

s equation solves for tension2.
now I must solve for ↑tension3↓,
given ↑tension2,tension1,ma,mb,mu↓
now trying to solve for tension3 without introducing any unknowns.
uck - I will now accept unknowns in solving for tension3.
reaction1 be the reaction of table in direction 90.

ation- 4: -tension3=0
ned by resolving forces at 180 to the horizontal for b

is equation solves for tension3 but introduces ↑reaction1↓.
now I must solve for ↑reaction1↓,
given ↑tension3,tension2,tension1,ma,mb,mu↓
now trying to solve for reaction1 without introducing any unknowns.
ation- 5: reaction1-mb*g=0
ned by resolving forces at 90 to the horizontal for b

is equation solves for reaction1.

ations extracted :
-ma*g+tension1=0
-tension1+tension2*cos(45)=0
tension1*cos(45)+(-tension2+tension3*cos(45))=0
-tension3=0
reaction1-mb*g=0

References

Annett J. (1966)
"Programmed Learning " in
"New Horizons in Psychology 1"
ed: B.Foss
Penguin Books
Middx

Barr A., Beard M. and Atkinson R. (1976)
"The Computer as a Tutorial Laboratory: the Stanford BIP Project"
International Journal of Man-Machine Studies
Vol 8, 567-596

Boyle J. and Anderson J. (1978)
"A CAL Approach to Population Dynamics"
International Journal of Mathematics Education in Science and Technology
Vol 9 No4, 467-476

Anderson J., Boyle C. and Yost G. (1985)

"The Geometry Tutor"

Proceedings of the 1985 International Joint Conference

for Artificial Intelligence

Briggs J. (1982)

"Teaching Mathematics with PROLOG"

BSc Thesis

Department of Computing

Imperial College

London

Brown J. and Burton R. (1978)

"Diagnostic Models for Procedural Bugs in Basic

Mathematical Skills"

Cognitive Science

Vol 2, 155-192

Brown J., Burton R. and deKleer J. (1982)

"Pedagogical, Natural Language, and Knowledge Engineering

Techniques in SOPHIE I, II, and III"

in: "Intelligent Tutoring Systems"

ed: D. Sleeman and J. Brown

Academic Press

New York

Brown J. and Van Lehn K. (1980)

"Repair Theory: a Generative Theory of Bugs in Procedural Skills"

Cognitive Science

Vol 4, 379-426

Buist G. (1978)

"Computer-Assisted Learning in the University of Surrey Chemistry Department"

International Journal of Mathematical Education in Science and Technology

Vol 9 No 4, 477-484

Bunderson C. (1974)

"The Design and Production of Learner-Controlled Courseware for the TICCIT system: a Progress Report"

International Journal of Man-Machine Studies

Vol 6, 479-491

Bundy A. (1982)

"The Nature of AI, a reply to Ohlsson"

Quarterly Journal of the SSAISB

Vol 47, 24-25

Bundy A., Byrd L., Luger G., Mellish C. and Palmer M. (1979)
"Solving Mechanics Problems Using Meta-Level Inference"
Proceedings of the 6th International Joint Conference in
Artificial Intelligence
Tokyo

Bundy A., Byrd L. and Mellish C. (1982)
"Special Purpose, but Domain Independent, Inference
Mechanisms"
DAI Research Paper No 179
Department of Artificial Intelligence
Edinburgh University

Bundy A., duBoulay B., Howe J. and Plotkin G. (1982)
"The Researcher's Bible"
DAI Occasional Paper No10
Department of Artificial Intelligence
Edinburgh University
Edinburgh

Burton R. and Brown J. (1982)

"An Investigation of Computer Coaching for Informal Learning Activities"

in: "Intelligent Tutoring Systems"

ed: D. Sleeman and J. Brown

Academic Press

New York

Chi M., Feltovich P. and Glaser R. (1981)

"Categorization and Representation of Physics Problems by Experts and Novices"

Cognitive Science

Vol 5, 121-152

Davis R. and King J. (1977)

"An Overview of Production Systems"

Machine Intelligence No 8

deGroot A. (1978)

"Thought and Choice in Chess"

Mouton, New York

Farrell R., Anderson J. and Reiser B. (1984)

"An Interactive Computer-Based Tutor for LISP"

Proceedings of the 1984 Conference of the American

Association for Artificial Intelligence

Ghesquiere J., Davies P. and Thompson C. (1974)

"Introduction to Tutor"

Plato Services Organisation

CERL

Goldenberg E. (1971)

"Special Technology for Special Children"

University Park Press

Baltimore

Goldstein I. (1982)

"The Genetic Graph: a Representation for the Evolution of

Procedural Knowledge"

in: "Intelligent Tutoring Systems"

ed: D. Sleeman and J. Brown

Academic Press

New York

Hinsley D., Hayes J. and Simon H. (1977)

"From Words to Equations: Meaning and Representation in Algebra Word Problems"

in: "Cognitive Processes in comprehension"

eds: Just and Carpenter

Lawrence Erlbaum Associates, Hillsdale, NJ

Hooper R. (1975)

"Two Years On, The National Development Programme in Computer Assisted Learning: Final Report of the Director"

Council for Educational Technology

Howe J., O'Shea T., and Plane F. (1980)

"Teaching Mathematics through Logo Programming: an Evaluation Study"

in: "Computer Assisted Learning: Scope, Progress and Limits"

ed: R. Lewis and E. Tagg

North Holland

Amsterdam

Howe J., Ross P., Johnson K., Plane F. and Inglis R. (1982)
"Learning Mathematics Through LOGO Programming: the Transition from Laboratory to Classroom"
DAI Working Paper No 118
Department of Artificial Intelligence
Edinburgh University
Edinburgh

IBM Corporation (1971)
"Coursewriter III, Version 3, Author's Guide"
IBM Corporation
New York

Jansweijer W., Elshout J. and Wielinga B. (1986)
"The Expertise of Novice Problem-Solvers"
Proceedings of the 7th European Conference on Artificial Intelligence
Brighton, UK

Johnson-Laird P. (1983)
"Mental Models"
Cambridge University Press

Jones M. and Tuggle F. (1979)

"Inducing Explanations for Errors in Computer-Assisted Instruction"

International Journal of Man-Machine Studies

Vol 11, 301-324

Kowalski R. (1984)

"Logic as a Computer Language for Children" in

"New Horizons in Educational Computing"

ed: M. Yazdani

Ellis Horwood

Chichester

Larkin J. (1977)

"Skilled Problem-Solving in Experts"

Technical Report, Group in Science and Mathematics Education

University of California

Berkeley

Larkin J. (1977)

"Skilled Problem-Solving in Physics: a Hierarchical Planning Model"

Technical Report, Group in Science and Mathematics Education,

University of California

Berkeley

Larkin J. (1977)

"Problem-Solving in Physics"

Technical Report, Group in Science and Mathematics Education

University of California

Berkeley

Larkin J. (1979)

"Skill Acquisition in Solving Physics Problems"

CIP No 409

Department of Psychology

Carnegie-Mellon University

Pittsburgh

Larkin J. (1979)

"Models of strategy for Solving Physics Problems"

Paper presented at the AERA Conference

San Francisco, CA

Larkin J. (1980)

"The Cognition of Learning Physics"

Applied Cognitive Psychology No 1

Department of Psychology

Carnegie-Mellon University

Pittsburgh

Larkin J., McDermott J., Simon D. and Simon H. (1979)

"Models of Competence in Solving Physics Problems"

CIP No 408

Department of Psychology

Carnegie-Mellon University

Pittsburgh

Larkin J., McDermott J., Simon D. and Simon H. (1979)
"Expert and Novice Performance in Solving Physics Problems"
CIP Working Paper No 410
Department of Psychology
Carnegie-Mellon University
Pittsburgh

Larkin J., McDermott J., Simon D. and Simon H. (1980)
"Expert and Novice Performance in Solving Physics Problems"
Science
Vol 208, 1335-1342

Larkin J., McDermott J., Simon D. and Simon H. (1980)
"Expert and Novice Performance in Solving Physics Problems"
Cognitive Science
Vol 4, 317-345

Larkin J. and Reif F. (1979)
"Understanding and Teaching Problem-Solving in Physics"
European Journal of Science Education Vol 1

Luger G. (1977)

"The Use of Protocols and the Representation of Semantic Information in Pulley Problems"

DAI Research Report No 36

Department of Artificial Intelligence

Edinburgh University

Luger G. (1981)

"Mathematical Model Building in the Solution of Mechanics Problems: Human Protocols and the Human Trace"

Cognitive Science

Vol 5, 55-77

Marples D. (1974)

"Argument and Technique in the Solution of Problems in Mechanics and Electricity"

CU ED/C-Educ/TRI

Department of Engineering

University of Cambridge

Marr D. (1977)

"Artificial Intelligence- A Personal View"

Artificial Intelligence No 9, 37-48

Matz M. (1982)

"Towards a Computational Theory of Algebraic Competence"

Journal of Mathematical Behaviour

Vol 3, No 1, 93-166

McDermott J. and Larkin J. (1978)

"Re-representing Textbook Physics Problems"

The Canadian Society for Computational Studies of

Intelligence

University of Toronto Press

Murphy P. (1981)

"Biological Simulations in Distance Learning" in

"Computer Assisted Learning: Selected Papers from the CAL

81 Symposium"

ed: P.Smith

Pergamon Press

Oxford

Newell A. and Simon H. (1972)

"Human Problem-Solving"

Prentice-Hall

Ohlsson S. (1982)

"Tell Me Your Problems: a Psychologist Visits AAAI82"

Quarterly Journal of the SSAISB

Vol 46, 27-31

O'Shea T. and Self J. (1983)

"Learning and Teaching with Computers"

Harvester Press

Brighton

Paige G. and Simon H. (1966)

"Cognitive Processes in Solving Algebra Word Problems"

in: "Problem-Solving: Research, Method, and Theory"

ed: B.Kleinmutz

John Wiley, NY

Palmer B. and Oldehoeft A. (1975)

"The Design of an Instructional System Based on Problem Generators"

International Journal of Man-Machine Studies

Vol 7, 249-271

Papert S. (1980)

"Mindstorms- Children, Computers and Powerful Ideas"

Basic Books

New York

Papert S. (1984)

"Tomorrow's Classrooms" in

"New Horizons in Education"

ed: M.Yazdani

Ellis Horwood

Chichester

Papert S., Watt D., diSessa A. and Weir S. (1974)

"Final Report of the Brookline LOGO Project"

AI Lab Memo 545

MIT

Massachusetts

Piaget J. (1971)

"The Science of Education and the Psychology of the Child"

Viking Press

Priest A. (1986)

"Solving Problems in Newtonian Mechanics"

Instructional Science

Vol 14, 339-355

Priest A. (1987)

"The Generation of Errors in Physics Problem-Solving"

Departmental Research Report No 9

Department of Computing and Mathematical Sciences

Oxford Polytechnic

Priest A. and Young R. M. (1986)

"Methods for Evaluating Micro-Theory Systems"

Proceedings of the Intelligent Computer-Assisted

Instruction Workshop

Windermere, Cumbria

Priest A. and Lindsay R. (1986)

"Algorithmic Processes in Physics Problem-Solving: An

Experimental Investigation"

Departmental Research Report No 7

Department of Mathematics, Statistics and Computing

Oxford Polytechnic

PLATO IV Software Group (1974)

"PLATO IV Authoring"

International Journal of Man-Machine Studies

Vol 6, 445-463

Reif F., Larkin J. and Brackett G. (1976)

"Teaching General Learning and Problem-Solving Skills"

American Journal of Physics Vol 4

Ritchie G. and Hanna F. (1984)

"AM: A Case Study in AI Methodology"

Artificial Intelligence

Vol 23, 249-268

Scanlon E., Hawkridge C. and O'Shea T. (1983)

"Modelling Physics Problem-Solving: Scripts and Production Rules"

Open University CAL Technical Report No 36

Self J. (1974)

"Student Models in Computer-Aided Instruction"

International Journal of Man-Machine Studies

Vol 6, 261-276

Skinner B. (1954)

"The Science of Learning and the Art of Teaching"

Harvard Educational Review

Vol 24 No 2

Slack J. (1984)

"The Birth of a New Discipline"

in: "Artificial Intelligence- Tools , Techniques and Applications"

eds: T. O'Shea and M. Eisenstadt

Harper and Row, NY

Sleeman D. (1981)

"Modelling Human Problem-Solving"

Artificial Intelligence

Vol 16

Sleeman D. and Brown J. (1982)

"Intelligent Tutoring Systems"

Academic Press

New York

Tait K., Hartley J.R. and Anderson R.C. (1973)

"Feedback Procedures in Computer-Assisted Arithmetic Instruction"

British Journal of Educational Psychology

Vol 43, 161-171

VanLehn K. (1981)

"On the Representation of Procedures in Repair Theory"

Cognitive and Instructional Sciences Series

CIS-16 (SSL-81-7)

Xerox Palo Alto Research Center

Palo Alto

California

Winston P. (1977)

"Artificial Intelligence"

Addison-Wesley

Winston P. (1984)

"Artificial Intelligence"

Addison-Wesley

Young R. M. and O'Shea T. (1981)

"Errors in Children's Subtraction"

Cognitive Science

Vol 5, 153-177